OS/2 Only!

Das fortlaufende Sammelwerk nur für OS/2

Band 1, Ausgabe 12/98, 1. Jahrgang

- **Neuigkeiten:**
  IBMs OS/2-Strategie
- **Hardware:**
  ZIP-Laufwerk von Iomega
- **Software:**
  OS/2-Systemtools
- **Konfigurieren und Anwenden:**
  OS/2-Ressourcenbedarf optimieren
- **Programmierung:**
  Workshop - DLLs
- **Netzwerk:**
  HTML-Workshop, Teil I
- **Großkunden:**
  ASTRAC Visualizer

CEF C.-E. Fischer
Buchverlag, Stuttgart

OS/2 Only!
Das fortlaufende Sammelwerk für OS/2

OS/2 Only!
Das fortlaufende Sammelwerk für OS/2

Bd.1, Ausgabe 12/98
1. Jahrgang

Impressum

Die *OS/2 Only!* wird herausgegeben vom C.-E. Fischer Buchverlag, Stuttgart.
Erscheinungsweise: Zweimonatlich, jeweils zum Monatsersten.
Anzeigen- u. Redaktionsschluß: Jeweils der 18. eines Monats vor dem Erscheinungsmonat.

Kontaktmöglichkeiten:
C.-E. Fischer Buchverlag
Bereich: OS/2 Only!
Wegländerstr. 24
D-70563 Stuttgart (Vaihingen)
Tel.: 0711-7802260
Fax: 0711-7802261
eMail: cefischer@t-online.de
WWW: http://www.cefischer.de

Abonnementverkauf und Anzeigenaufnahme: Thomas Ropielewski
Lektorat: Carsten Momberg
Karikaturen: Ina Becker
Verantwortlich für den Inhalt: Clemens-Elias Fischer
Reproduktion: Schwabenrepro GmbH, Stuttgart
Druck u. Verarbeitung: Druckerei Jauch GmbH, Stuttgart

1. Aufl., 1998
Printed in Germany

ISSN 1436-8307

Inhalt

Hinweise zum Gebrauch

Im folgenden Abschnitt finden Sie Hinweise zum Gebrauch unseres Sammelwerkes. Arbeiten Sie mit der *OS/2 Only!* zum ersten Mal, sollten Sie diesen Abschnitt aufmerksam durchlesen. Sind Sie mit dem Gebrauch des Magazins vertraut, möchten wir Ihnen empfehlen, immer mal wieder einen Blick auf diese Hinweise zu werfen, da sich von Zeit zu Zeit etwas ändern kann.

Grundsätzlicher Aufbau

Die *OS/2 Only!* gliedert sich in der Regel wie folgt:

- Editorial
- Neuigkeiten

  Mit einem Schwerpunktthema und den Abschnitten »Kurzmeldungen, Bezugspunkte und Kontakte« sowie »Leserbriefe«, wenn diese dem Verlag vor Redaktionsschluß vorliegen.

- Hardware

  Mit den Abschnitten: Theorie und Praxis. Außerdem (jedoch nur in jeder 2. Ausgabe beginnend bei Band 2): Eine umfassende Liste mit Fundorten für Treiber und eine Listung OS/2-tauglicher Hardware. Die Listen werden in jeder 2. Ausgabe erweitert bzw. korrigiert.

- Software

  Mit einem Schwerpunktthema und den Abschnitten Share- und Freeware sowie einem Anwendungsworkshop. Außerdem (jedoch nur in jeder 2. Ausgabe beginnend bei Band 2): Eine umfassende Liste mit nativer OS/2-Software, deren Hersteller und Händler. Mit einer kurzen Beschreibung zu jedem Produkt. Die Listen werden in jeder 2. Ausgabe erweitert bzw. korrigiert.

- Know how

  Mit einem Schwerpunktthema und den Abschnitten: Tips & Tricks, OS/2-Basiswissen.

- Programmieren

  Mit einem Schwerpunktthema oder einem Workshop und den Abschnitten: Java, C/ C++, Pascal und REXX.

- Multimedia

  Mit einem Schwerpunktthema und den Abschnitten: Hardware, Software.

- Netzwerk

  Mit einem Schwerpunktthema in Theorie und/ oder Praxis und den Abschnitten: DFÜ, Internet, TCP/IP.

- Großkunden

  Mit den Abschnitten: Firmenprofil, Produkte.

- In eigener Sache

  Mit den Abschnitten: Druckwerke, Software, Besondere Angebote.

- Anhang, Verzeichnisse, Lexikon.

Von Ausgabe zu Ausgabe können die Abschnitte einzelner Kapitel in ihrem

Umfang variieren oder auch ganz ausbleiben.

Software und Treiber
In Artikeln genannte Software wird, sofern möglich, stets zu jeder Ausgabe auf unserer Homepage zur Verfügung gestellt. Das gilt auch für benötigte Treiber.

Satz und Typographie
In der Kopfzeile einer jeden Seite finden Sie das jeweilige Kapitel und dessen Abschnitt wieder, in dem Sie sich gerade befinden. Kapitelname und Bezeichnung des Abschnitts sind mit einem Doppelpunkt voneinander getrennt. Alle Abschnitte sind im im Inhaltsverzeichnis wiedergegeben.
Tabellen und/ oder Abbildungen sind jeweils innerhalb eines Artikels numeriert. Eine Auflistung aller Abbildungen und Tabellen eines Bandes finden Sie im Abbildungsverzeichnis am Ende eines jeden Bandes und zwar einmal alphabetisch nach Titeln und außerdem thematisch geordnet.
Für das Suchen nach bestimmten Themen können Sie auch das Stichwortverzeichnis benutzen, das nach Stichworten ebenfalls sowohl alphabetisch als auch thematisch geordnet angelegt ist.

Artikel beginnen in einem Kapitel mit einer **fettgedruckten** Überschrift, die etwas größer als der normale Fließtext ist. Einzelne Abschnitte eines Artikels sind vom jeweils vorangegangenen Abschnitt mit einer Leerzeile getrennt und werden mit einer *kursiven* oder einer ***fettkursiven*** Überschrift eingeleitet.

Im Text selbst finden Sie Produktnamen, Herstellerbezeichnungen usw., die gerade angesprochen werden, *kursiv* gedruckt, also etwa: Das *ZIP-100* von *Iomega*.
Bestimmte Bezeichnungen, vor allem technische Ausdrücke, können **fett und unterstrichen** dargestellt sein. In einem solchen Fall können Sie den auf diese Weise angezeigten Begriff im Lexikon nachschlagen, das Sie jeweils am Ende eines Bandes finden. Die einzelnen in einer Ausgabe gekennzeichneten Begriffe sind im lexikalischen Teil alphabetisch geordnet.

❑ Aufzählungen werden mit dem Symbol ❑ eingeleitet. Numerierungen erfolgen mit arabischen Ziffern. In Numerierungen eingebettete Aufzählung werden mit einem **o** kenntlich gemacht.

! ***Wichtige Abschnitte, etwa besondere Hinweise, werden durchgängig fett und kuriv gedruckt und mit einem Ausrufezeichen versehen.***

Die einzelne Schritte einer Vorgehensweise werden mit den Zahlensymbolen ❶ ❷ ❸ ❹ ❺ ❻ usw. geordnet und kenntlich gemacht.

Tip: *Ein besonderer Tip zur Konfiguration bzw. Handhabung eines bestimmten Programms oder einer Hardwarekomponente, wird mit einem kursiv gedrukken, großen* ***Tip*** *hervorgehoben. Der folgende Text wird außerdem kursiv wiedergegeben.*

Es gibt viel zu tun...

Lange hat es gedauert, bis die letzten Planungsarbeiten an unserem neuen OS/2-Zeitschriftenprojekt abgeschlossen waren, und wieder mit erheblicher Verzögerung erschien endlich die Dezemberausgabe der *OS/2 Only!*, die Sie nun in den Händen halten; doch die Anfangshürden sind genommen, die zweite Ausgabe in Arbeit, und wir beginnen bereits mit der Zusammenstellung der ersten *OS/2 Only!*-CD, die Ihnen Treiber, Schriften, Software und viele interessante Informationen bieten wird. So möchte ich mich als Verlagsleiter zu aller Anfang bei Ihnen für Ihre Ideen, Ihr Interesse und letztlich natürlich Ihre Geduld recht herzlich bedanken. Aus gutem Grund.

Die OS/2-Gemeinde ist schon zu recht verunsichert, wenn es um Ankündigungen und dergleichen geht. So mancher Plan um OS/2 wird rasch auf Eis gelegt, ehe er überhaupt annähernd feste Formen angenommen hat, und so sieht sich der OS/2ler -bzw. die OS/2lerin- mit einer Enttäuschung nach der anderen konfrontiert. »Das Projekt ist wohl schon gestorben«, hörten wir in den letzten Monaten immer wieder. Nein, das ist es nicht! Mit der *OS/2 Only!* zeigen wir Ihnen, daß wir nicht nur planen - sondern auch umsetzen.

So richten wir unsere Politik im Laufe des nächsten Jahres ganz auf OS/2 aus und haben uns ein ehrgeiziges Ziel gesetzt, das, sorgsam durchdacht, ebenso umgesetzt werden kann, wie diese Zeitschrift: Im Zusammenhang mit der *OS/2 Only!* und unserer Software bieten wir Ihnen eine besondere Form des Supports, die Ihnen sicher zusagen wird. Mehr darüber lesen Sie im Abschnitt *In eigener Sache*.

Diese Ausgabe setzt einen Anfang. Vieles soll noch folgen, das wir aus Zeitgründen nicht in der Erstausgabe unterbringen konnten. Das beginnt bei Verbesserungen eher oberflächlicher Natur und reicht bis zu einem noch reicheren Informationsangebot. Ihre Meinung und Ihr Wissen sind uns wichtig, und nur mit Ihrem Feedback können wir das Magazin optimieren. Schreiben Sie uns also! Egal ob es Tips & Tricks sind, Artikel, kurze Neuigkeiten, Veranstaltungstermine, Hinweise zu Vereinen, Webadressen, Ansprechpartner, Beispielprogramme, Erfahrungen mit Hard- und Software, ganze Workshops: Hauptsache Ihr Beitrag bezieht sich auf OS/2! Wenn Sie Ihre Erfahrungen weitergeben wollen, scheuen Sie sich nicht, uns Ihre Beiträge zu schicken. Die OS/2-Anwender brauchen ein Forum: Hier ist es! Sie müssen es nur nutzen. Und: Sagen Sie uns, was Ihnen auf dem Herzen liegt, wenn Ihnen etwas nicht paßt, was verbessert werden könnte, und was beibehalten werden soll.

Einiges bleibt aber auch ganz sicher so, wie es jetzt ist: Etwa die Erscheinungsweise und die grundsätzliche Aufmachung: Wir verzichteten auf teuren »Schnickschnack«, aufwendige Layoutgestaltungen und farbige Abbildungen sinnloser Piktogramme. Statt dessen setzen wir auf einen schlichten Stil und eine übersichtliche Gestaltungen, die wir, so hoffe ich, auch realisieren konnten. Information soll im Vordergrund stehen. Somit besitzt diese Zeitschrift auch einen gewis-

sen Buchcharakter, und ich hoffe, dies gefällt Ihnen.
Sie finden in jeder Ausgabe der *OS/2 Only!* stets Angebote unterschiedlicher Hersteller und natürlich auch von uns. Wir haben für Sie zum Beispiel unsere Abonnementsangebote und Informationen zur Zukunft unseres OS/2-Programms in diese Ausgabe aufgenommen. Was sich bei den einzelnen Herstellern und bei uns so tut, werden Sie also nicht verpassen. Die erste ausführliche Herstellerliste für Hard- und Software inkl. der notwendigen Kontaktadressen finden Sie in Band 2 der *OS/2 Only!* Diese Listen werden in jeder zweiten Ausgabe aktualisiert bzw. erweitert, um Ihnen einen möglichst umfassenden Marktüberblick zu geben. Mit der ersten Händlerliste erscheint auch die erste Treiberliste, die in gleicher Weise auf dem neuesten Stand gehalten wird.

Das Gesamtkonzept der *OS/2 Only!* ist etwas anders, als das anderer Fachzeitschriten. Das heißt, daß Workshops und umfangreiche Artikel im Vordergrund des Magazins stehen. Ziel ist, daß Sie so umfassend wie nur irgend möglich informiert werden. Ich selbst hasse Fachliteratur, die neugierig macht, aber tiefergehende Fragen unbeantwortet läßt. Dabei soll es ganz egal sein, ob wir Netzwerklösungen (wie im Band 2 und 3) betrachten oder Programmierprobleme: Detailfülle ist uns wichtig, Sie sollen mit dem neuen Magazin so viel wie möglich anfangen können. Nicht zuletzt wollen wir den aktuellen Aspekt einer Zeitschrift mit der Ausführlichkeit eines Buches verbinden und damit v.a. auch die schlechte Situation deutschsparachiger Litertur zu OS/2 zu verbessern und manche peinliche Lücke zu schließen helfen. Im Kapitel *In eigener Sache* finden Sie die Vorschau auf die Themen der nächsten drei Ausgaben. Ihre Beiträge können wir aber jederzeit in eine Ausgabe aufnehmen, sofern Sie die Zeiten des Redaktionsschlusses beachten (genug Platz ist da). Zusammen müßte es uns möglich sein, allen OS/2-Anwendern-das zu bieten, was so dringend fehlt: Information.

Bei unserer Arbeit lassen wir uns auch nicht von all jenen beirren, für die OS/2 kein Thema mehr ist. Ich arbeite seit der Version 2.1 mit OS/2 und sehe keinen Grund, auf ein anderes System umzusteigen, vor allem wüßte ich auch gar nicht auf welches.
Was andere also auch sagen: Wir haben ein vorzügliches Betriebssystem, lassen Sie sich nicht verunsichern. Wir werden Ihnen in naher Zukunft interessante Lösungen für OS/2 anbieten, Sie regelmäßig informieren, Tips aus allen relevanten Bereichen geben und uns für den Kreis der Privatanwender engagieren - um Unterstützung zu bieten, die man nicht mehr oder bislang noch nicht fand. Den Grundstein setzen wir mit der *»Only!«*. Und wir freuen uns, daß Sie dabei sind!

Ihr

Clemens-Elias Fischer
(Verlagsleiter)

Neue Wege und Visionen

Das waren Zeiten! Als IBM mit Paulchen Panther für OS/2 2.1 warb, und einem hübsche Broschüren mit verführerischen Angeboten ins Haus flatterten (wenn man was verkaufen will, weiß man eben, was den Leuten gefällt). Auch mit dem Erscheinen von Warp 3 schien sich etwas auf dem Markt der Betriebssysteme zu verändern. Sicher kann man sich noch an die Werbeslogans erinnern und die kurzen Spots, die im Fernsehen zu sehen waren. Durch die aggressivere Marktstrategie der IBM bekam die Treiberunterstützung mit Erscheinen von Warp 3 auch einen Aufschwung, ebenso interessierten sich die Softwarehersteller für OS/2. Viele Anwendungen, mit denen man heute arbeitet, erhielten in dieser Zeit entscheidende Qualitätssteigerungen. Geschwindigkeit, Stabilität und Kompatibilität waren die wichtigsten Punkte, die so manchen Windows-Anwender tatsächlich auf OS/2 umstiegen ließen. Man hätte meinen können, IBM wollte es mit Microsoft aufnehmen, so wie da der Endkunde angesprochen wurde, und das war wohl auch tatsächlich der Fall. Ist das nun immer noch so? Und überhaupt: Was passiert denn nun mit dem Produkt? Wir haben für Sie einige Neuigkeiten gesammelt, Fragen geklärt und uns Informationen von IBM näher für Sie angesehen.

Keine Produktwerbung mehr!

Eine der ersten Aussagen, die wir hörten; was nicht heißt, daß IBM keine Werbung mehr betreiben würde, aber die Inhalte haben sich entschieden geändert. Eine Produktwerbung, wie man sie kannte, werde es nicht mehr geben, vielmehr werde produktübergreifend geworben und zwar für IBMs Vorstellungen um eine moderne Welt und das Mittel, sie zu vernetzen: *Network Computing*. Oder für Produkte, die mehr als andere eine Rolle in diesem Konzept spielen, etwa *Lotus Notes*. Ansonsten wird von Plattformen generell keine Rede mehr sein.

Das hat für viele den Anschein, IBM sei am Verkauf von OS/2 wenig interessiert. Nun richtete sich die alte Produktwerbung aber sicherlich an die breite Masse, kaum an einen auserlesenen Kundenkreis. IBM sieht seine wichtigsten Abnehmer aber im Bereich der Großanwender, also in all jenen an der Spitze des Marktes, womit eine Publikumswerbung von sich aus unsinnig wird. Und was nun OS/2 im speziellen betrifft: Das Betriebssystem nimmt auch eine ganze andere Stellung ein. Einst als Alternative zu MS-Windows ins Feld geführt, scheint es jetzt für IBM ein optimales Trittbrett zu sein, um OS/2-Kundenkreisen zu ermöglichen, auf den mehr oder minder schnell anfahrenden *Network Computing*-Zug aufzuspringen.

Network Computing?

Dieser Begriff ist schon mehrere Male gefallen. Zugegeben, eine genaue Definition dafür zu finden, ist nicht einfach, und so mancher fragt sich, was das eigentlich sein soll. Im großen und ganzen beinhaltet es zwei Punkte:

- Auf Serverlösungen basierende Netzwerke und
- das Durchsetzen von Java als neuen Applikationsstandard.

Der erste Punkt gibt im wesentlichen die Marktausrichtung der IBM wieder: Weder der Privatanwender, noch der Betreiber eines kleinen Netzwerkes ist an einer serverbasierenden Lösung interessiert. Für große Netzwerkumgebungen sieht das aber anders aus, so daß es nur logisch ist, warum IBM auf dieses Konzept setzt.

Java hat für IBM vor allem wegen seiner Plattformunabhängigkeit große Bedeutung. Auf diese Weise soll der Kunde nicht gezwungen werden, ein bestimmtes Betriebssystem mit spezifischen Anwendungen zu kaufen. IBM versucht mit diesem plattformübergreifenden Konzept zweifellos, der Microsoftdominanz entgegenzutreten. Das eigene Betriebssystem in breiteren Marktsegmenten als in denen der Großkunden durchzusetzen, mißlang offensichtlich, was aber weniger am Produkt, sondern vielmehr am Marketing seines Herstellers und an den festgefahrenen Verhältnissen des EDV-Marktes lag. So hat IBMs Unterstützung von »Heterogenität« und Plattformunabhängigkeit den Anschein, als wolle man das »MS-Problem« von einer anderen Seite angreifen: *No Windows! Pure Java!* - was sich allerdings ein wenig selbst ad absurdum führt: Sollte Java wirklich einmal ein Applikationsstandard werden, was mal dahingestellt sein soll, zumindest uns scheint das sehr schwer vorstellbar, dann kann der OS/2-Benutzer natürlich getrost bei seinem Warp bleiben. Aber es gibt auch keinen triftigen Grund, sich von Microsoft zu entfernen, bzw. auch keinen Anlaß für IBM, die eigene Hardware nach wie vor mit Microsoft-Produkten zu vertreiben: Wenn Java plattformunabhängig ist, läuft es ja gerade *auch* auf Windows. Daß die Java-Unterstützung von Warp vorbildlich ist, wird wohl kaum ein Grund für den Windowsanwender sein, deswegen auf OS/2 umzusteigen. Überhaupt: Ein Windowsanwender schreibt seine Texte so und so mit Windows[1], nicht mit Java. Und wenn er doch eine erstklassige Java-Unterstützung haben möchte, dann kann er gar nicht auf OS/2 umsteigen, weil er ja nicht weiß, was das überhaupt ist. Also doch den Endanwender ansprechen? Nein, nein: Die Spitze des Marktes ist ohnehin viel rentabler; dort werden Standards geschaffen, und der Volksmund irrt, wenn er meint, Kleinvieh mache auch Mist!? Ob die IBM-Rechnung also wirklich aufgeht, bleibt fraglich. Profitieren kann der kleine OS/2-Anwender aber auf jeden Fall davon.

Java als Zugpferd

IBM setzt ganz auf Java und empfiehlt, sich völlig auf die Entwicklung von Java-Programmen zu konzentrieren, ferner diesem Trend den Vorzug gegenüber plattformabhängigen Entwicklungsstrategien zu geben.

Aus diesem Grund kann sich die Java-Implementierung, die man in OS/2 findet, auch durchaus sehen lassen. Warum Java von so großer Bedeutung ist, haben wir bereits erwähnt. Daher bietet IBM auch

---

1 Was natürlich unsinnig ist, mit Windows allein läßt sich gar nichts schreiben. Aber das ist die verbreitete Meinung in großen Anwenderkreisen: »Meine Texte schreibe ich doch mit Windows.« Etwas anderes kennt man eben nicht und braucht man auch nicht. Die Macht der Werbung...

Support für die Portierung von OS/2-Programmen nach Java, was nicht verwunderlich erscheinen sollte. Ein so ganz optimales Pferd scheint Java allerdings auch nicht zu sein, jedenfalls noch nicht. Als von Java zum ersten Mal zu hören war, schien die Welt ja regelrecht begeistert gewesen zu sein. Wahrscheinlich jedoch schrie die Presse mal wieder lauter als es eigentlich beabsichtigt war, denn der große Umschwung der Softwarehersteller auf Java blieb aus.

Wenn IBM davon spricht, daß der Gebrauch von Java zur Erstellung bedeutender Anwendungen im nächsten Jahr um 56 % steigen wird, dann scheint dies ebenso unrealistisch. Microsoft etwa würde eine Java-Office-Suite sicher nicht gefallen und wenn sogar IBM selbst eingestehen muß, daß Microsoft nach wie vor der größte und einflußreichste Konkurrent auf dem Markt ist, dann wird es so etwas auch nicht so rasch geben. Daß Java genutzt wird, ist klar und daß dieser Prozentsatz ebenfalls steigen wird, ist auch wahrscheinlich. Aber die Softwarehersteller ziehen immer noch nicht am Java-Strang. Das ist leicht einzusehen: Der Umstieg auf Java würde so manchen Hersteller enorm viel kosten. Es wäre unrentabel, vorhandene Module und Bibliotheken in eine Sprache zu überführen, welche als Applets auf irgend einer Plattform ohnehin nie die gleiche Leistung erbringen können, als ein natives Programm. Und: Wozu Java, wenn man doch Windows hat? Steht einem anderen Betriebssystem dieses Fensterprogramm im Weg, dann einem sogar plattformunabhängigen Konzept erst recht.

Dennoch bietet Java wegen seiner Plattformunabhängigkeit interessante Möglichkeiten und wird mittelfristig sicherlich an Bedeutung gewinnen, womit auch die Anzahl verfügbarer Anwendungen steigen wird; wenngleich auch die Großkunden, zumindest in Deutschland, keinen sonderlichen Gebrauch davon machen bzw. unbedingt machen werden. Hier laufen entsprechende netzwerktaugliche Branchenanwendungen ohnehin direkt auf dem **PM**, ohne irgend welche Zusätze oder Schnörkel. Daneben gibt es Branchen, in denen Echtzeitapplikationen benötigt werden, und für derlei Belange dürfte Java recht ungeeignet sein, wie man sich sicher vorstellen kann. Da gerade komplexe Programme in den oberen Marktregionen zu finden sind, denken wir nicht, daß Java aus Sicht der jetzigen Marktlage jemals ein Applikationsstandard werden kann. Native OS/2-Programme werden damit auch nicht so rasch an Bedeutung verlieren, und welcher Anwender würde einer 32-Bit-Applikation einem Java-Applet keinen Vorrang geben?

Natürlich wollen wir Java ganz bestimmt nicht ablehen. Es finden sich sehr wohl Bereiche, in denen sich ein Einsatz anbietet (z.B. wenn keine nativen OS/2-Programme vorhanden sind), und so werden wir auch über die Entwicklung von Java-Programmen ausführlich berichten; aber wir möchten nicht ganz dem Taumel um die Java-Euphorie verfallen. Java mag seine Vorzüge haben, aber es besitzt auch Nachteile. Und es kann durchaus sein, daß letztere im Trommelwirbel um den neuen Soll-Standard untergehen.

Was geschieht nun mit OS/2?

OS/2 ist ganz in das Konzept des *Network Computing* eingebettet:

- Die Entwicklung von OS/2 wird *nicht* eingestellt, was immer man auch sonst noch hört. Einen neuen Client (Warp 5) wird es wohl zugunsten der Weiterentwicklung von **WSoD (Work Space on Demand)** nicht mehr geben, dafür einen neuen Warp Server (Warp Server 5). Zwar kursieren Gerüchte, IBM wolle aufgrund des Drängens seitens bedeutender Kunden ein Warp 5 herausgeben; aber bestätigt wurde das nicht.

- Die **WPS** wird nicht weiterentwickelt. Allerdings wird sie in den FixPaks berücksichtigt werden, so daß man auf eine Korrektur der Oberfläche nicht verzichten muß. Mit Weiterentwicklung sind vielmehr tiefgreifende Änderungen an der Architektur der WPS gemeint.

- Sowohl für den aktuellen Warp Server *als auch* für die Clients (Warp 3 *und* Warp 4) wird IBM regelmäßig Fehlerkorrekturen und dergleichen-bieten.

Man braucht also nicht zu fürchten, der bestehende Client erführe keine Verbesserungen mehr. Gerade bei der Vorbereitung dieser Ausgabe haben wir gemerkt, wie sehr ein bestimmter Bereich verbessert wurde. Codeupdates wird es, wie man es bereits kennt, mit FixPaks geben. Daneben bietet IBM:

- Verbesserungen der *JVM (Java Virtual Machine)* hinsichtlich Funktionalität und Performance, um die Verwendung zukünftiger Java-Versionen auf OS/2 zu gewährleisten

- auch in Zukunft die Netscapeprodukte (*Navigator, Communicator*) für OS/2

- die Einbindung neuer Hard- und Softwarestandards in die bestehenden Client- und Serverprodukte

- Verbesserungen der Hardwareunterstützung von OS/2.

Damit sieht es um OS/2 gar nicht so schlecht aus, darf man dem Hersteller Glauben schenken. Die starke Ausrichtung auf Java und Codeverbesserungen an den verfügbaren Clients geben weder einen Grund zu behaupten, OS/2 werde nicht mehr weiterentwickelt, noch es sei unmodern oder folge nicht den Trends des Marktes. Dem Endanwender kommt das ebenfalls zugute, wenn man bedenkt, daß man nach wie vor mit einem *Warp 4 Client* oder mit *Warp 3 Connect* ein ganzes TCP/IP- oder Peer-Netzwerk aufbauen kann, ohne daß von serverbasierenden Lösungen die Rede wäre.
Daß sich der Clientcode via FixPaks kontinuierlich verbessern läßt (zumindest meistens), ist altbekannt, und nahezu jeder OS/2-Anwender macht davon Gebrauch. Da dies der IBM-Upgrade-Weg ist und nicht das ständige Verbreiten neuer Releases, braucht man sich auch nicht verunsichert zu fühlen, weil es keine neue Client-Version geben wird. Eine Verbesserung der Codebasis und deren Erweiterung findet auch so statt; und wenn man zusätzlich bedenkt, wie technisch ausgereift OS/2 im Gegensatz zu gewissen Konkurrenzprodukten ist, muß man auch nicht von einer Version zur anderen hetzen: Entwicklung kann auch im stillen stattfinden.

Die fehlende Weiterentwicklung der WPS ist eine Konsequenz aus der Integration von OS/2 in das Konzept des *Network Computing*: Die WPS wird als Benutzeroberfläche in großen Netzwerkumgebungen ohnehin kaum verwendet und findet ihre größte Akzeptanz im Kreis der Endanwender. Da IBM *WSoD* als netzwerkgeeignetes Interface favorisiert, bleibt die WPS als Ansammlung von Dateien im Client zurück (wenngleich diese auch aktualisiert werden, zumindest noch). Daneben hat sich die WPS nicht als eine Architektur erwiesen, die bei den Softwareherstellern auf breite Akzeptanz stieß. Wirkliche WPS-Anwendungen sind nach wie vor rar und werden es auch bleiben. Für den Privatanwender ist eine fehlende Weiterentwicklung der WPS allerdings ärgerlich, da die Oberfläche viele grundsätzliche Mängel aufweist, die nicht mehr vom Hersteller behoben werden. Hier ist der Anwender nach wie vor auf Zusatzprodukte angewiesen, um die Funktionalität der WPS zu erhöhen und Schwächen auszugleichen.

Ausblicke für die Kleinen

Wenn auch die ein oder andere Tatsache, die durch die Verlagerung von OS/2 im Zeichen des *Network Computing* entsteht, für den Privatanwender bitter erscheinen mag, so fällt ein Blick in die Zukunft auch für die Marktregion, in der wir uns bewegen, doch recht moderat aus.

Ein bedeuetendes Probleme dürfte natürlich »mangelnde« Unterstützung sein, obgleich sich hier eigentlich auch nicht so viel ändert: Der OS/2-Anwender muß schon wissen, wo Treiber, Updates und entsprechende Programme zu bekommen sind, und das mußte er immer. Ganz allein steht er aber auch nicht, und es gibt wegen einer anderen IBM-Strategie keinen Grund, panisch zu werden.

Eine weitere Ursache zur Besorgnis ist für manchen Anwender auch die verquickte Situation, die entsteht, wenn es an nativer OS/2-Software mangelt. Allerdings wäre es marktverfremdend zu behaupten, kein Hersteller interessiere sich mehr für die Entwicklung von OS/2-Software. Programme gibt es nach wie vor, auch in neuen Versionen. Hier muß man ebenfalls wissen, wo man sie bekommen kann und was sie leisten. Vor allem im Bereich der Shareware-Szene findet man interessante Applikationen und viele Anwendungspakete sind auch für die OS/2-Plattform verfügbar.

Was letztlich IBM angeht: Die extreme Ausrichtung auf den Großkunden kann dem Endanwender letztlich nicht schaden. Schließlich kann es sich IBM als Hersteller nicht erlauben, Unternehmen, die OS/2-Investionen in Millionenhöhe getätigt haben, durch die Einstellung der Weiterentwicklung des Systems als solchem zu verlieren. Die immer wieder gehörten »OS/2 ist tot!«-Rufe dürften damit irrelevant sein, und das heißt, daß auch der Endanwender nicht zu fürchten braucht, sein Betriebssystem verschwände von der Bildfläche oder es gäbe gar keine Unterstützung neuer Hardware und Standards mehr.

Support gibt es also, man muß ihn nur zu nutzen wissen; und die Teile aus dem IBM-Angebot »herausfischen«, die für den weiteren Betrieb des Schreibtischcomputers und des kleinen Netzwerkes von Bedeutung sind. **(verl)**

Quelle: IBM Deutschland GmbH, Stuttgart

FixPaks fürs Betriebssystem

Sowohl für Warp 3 als auch Warp 4 gibt es wieder neue FixPaks. Warp 3-Benutzer können mit dem FixPak 38 ihr OS/2 auf den neuesten Stand bringen. Für Warp 4 steht das FixPak 8 zur Verfügung. Wir werden für die nächste Ausgabe das FP 38 für Warp 3 und FP 8 für Warp 4 auf unsere Rechner aufspielen, Ihnen die Verbesserungen vorstellen und Hinweise zur Installation geben.

Sie können sich die Dateien der FixPaks aus dem Internet ziehen, am besten vom IBM-FTP-Server *ftp.software.ibm.com.* Eine Bestellung über IBM ist ebenfalls möglich. Die OS/2-FixPaks sind auf den *ServicePac CDs* enthalten. Eine CD kostet im Einzelbezug DM 25,00 inkl. Versand und MwSt. Die *ServicePac CDs* erscheinen monatlich. Ein Jahresabo zu DM 230,00 inkl. Versand und MwSt. ist ebenfalls möglich, jedoch für den Privatanwender nur bedingt sinnvoll, da die *IBM ServicePac CDs* auch Korrekturen für DOS- und Windowsprogramme enthalten, und OS/2-Fixes sowie OS/2-Programme, die für den Endanwender von Nutzen sein dürften, nicht jeden Monat auf den CDs zu finden sind. In einer Datenbank unter *http://servicepac.mainz.ibm.de* können Sie allerdings nachschlagen, welche FixPaks auf welchen *ServicePac CDs* enthalten sind. Nur die CDs, die Sie wirk-

Fixes und Updates

Name	*Adresse*	*Hinweise*	*Wertung*
IBM Techn. Außendienst Mainz	http://servicepac.mainz.ibm.de	Anlaufpunkt für die IBM ServicePak-CDs. Bestellmöglichkeiten und Kontaktadressen, sowie eine Datenbank, welche die Inhaltsverzeichnisse alter CDs wiedergibt.	●●●
IBM Fixpaks, Treiber	http://ps.software.ibm.com	Eher eine Seite mit Links zum Wesentlichen. Allerdings sind auf den Seiten Informationen zu den FixPaks im allgemeinen verfügbar, die als Hintergrundwissen ganz brauchbar sind.	●
Fixes und Patches	http://service5.boulder.ibm.com/pspfixpk.nsf	Ein Zugriff auf Fixes und Patches aller IBM-Produkte über eine HTML-Seite. Wenn man weiß, was man benötigt, greift man jedoch lieber über FTP auf die Dateien zu.	●●●
Fixpaks über FTP	ftp.software.ibm.com/ps/products/os2/fixes	Hier kann man sich jeweils die aktuellen Fixpaks auf seinen Rechner laden. Login via anonymous FTP.	●●●●

lich brauchen, sollten Sie ordern. Das FP 38 für Warp 3 befindet sich auf der CD 12/98, das FP 8 für Warp 4 auf der CD 10/98. Die ServicePac CDs lohnen allerdings auch hinsichtlich anderer Fixes, Java oder der Netscape Produkte: Der Communicator 4.04ef (Deutsche Version) z.B. ist ebenfalls auf der CD 11/98 zu finden, ebenso wie das dazugehörige OS/2 Plug-in Pack.

Wir bieten ebenfalls die erforderlichen *ServicePac CDs* (10/98, 11/98, 12/98) zu einem Preis von DM 23,50 inkl. Versand und MwSt. Wir kummulieren dazu eingehende Aufträge. Die nächste Sammelbestellung wird am 19. Januar 1999 aufgegeben. Aufträge, die bis zum 19. Januar 8.00 Uhr bei uns eingehen, können berücksichtigt werden. *Sämtliche* Fehlerkorrekturen für OS/2 werden aber auch auf den halbjährlich erscheinenden *OS/2 Only!* CDs erhältlich sein (zum ersten Mal im Juni 1999).

Service Pac CDs über IBM bestellen

Einzelbestellungen:
Tel.: 01805 / 223399
Fax: 01805 / 223340

Jahresabonnement und Händler:
Tel.: 06131-845542
Fax: 06131 / 84-5526 / -6379

Ansprechpartner:
IBM Informationssysteme GmbH
Product Support Service / Kst. 7848
z.Hd. Peter Stey
Hechtsheimer Str.2
55131 Mainz
eMail: stey@de.ibm.com

Software Server

Name	*Adresse*	*Hinweise*	*Wertung*
Hobbes Archiv	http://hobbes.nmsu.edu/pub/os2	Softwarearchiv mit viel *aktueller* Share- und Freeware zu allen OS/2-relevanten Bereichen; Codebeispiele für nahezu alle Sprachen mit interessanten Tools; Treiber.	●●●●
LEO Link Everything Online	http://www.leo.org/archiv/software/os2	Softwarearchiv zu allen OS/2-relevanten Bereichen. Treiber, Codebeispiele, Laufzeitbibliotheken.	●●●
IBM EWS Software	ftp://ftp.pc.ibm.com/pub/pccbbs/os2_ews	Software von Angestellten der IBM. Zum Teil recht alt. Ein Besuch lohnt sich trotzdem. Es gibt viele nützliche EWS-Programme.	●●
OS/2 Super Site	http://www.os2ss.com	Archiv zu allen OS/2-relevanten Bereichen, teilweise jedoch nicht so umfassend und aktuell wie Hobbes. Dennoch brauchbar. Gute Anwender-Homepages	●●●

Software Server und News-Archive im Dezember

In unserem Software Server-Verzeichnis haben wir Server im Internet angegeben, die einen Besuch wert sind. Altgedienten OS/2-Anwendern werden die Adressen bekannt sein. Anwender, die auf Warp umsteigen, sollten sich dort zuerst umsehen, wenn sie nach Software und Treibern suchen. Wir geben grundsätzlich nur solche Server an, die wir uns auch angesehen haben, um sie bewerten zu können. Wir benoten Sie hinsichtlich ihres Softwareangebotes mit wenig brauchbar (●) bis sehr brauchbar (● ● ● ●). Newsarchive werden weniger nach ihrem Umfang als nach der Brauchbarkeit der Informationen bewertet, sowie deren Streuung (möglichst viele Themen sollen abgedeckt sein). Alle Einträge finden sich auch im Anhang unter Bezugspunkte und Kontakte. In diesem Kapitel werden stets nur Neuzugänge angegeben. Alle alten Einträge kommen in den Anhang. Der Anhang bleibt aktuell, d.h. die Adressen werden vor der Drucklegung einer jeden Ausgabe auf Änderungen überprüft. Wenn Sie Tips bezüglich guter Internetadressen haben, lassen Sie es uns wissen.

Neue Druckertreiber von EPSON

Wenn man sich genau erinnert, so war EPSON einer der wenigen Hersteller, der eigene OS/2-Treiber für seine Drukker bot. Eine Zeit lang war das nicht mehr so, denn EPSON Deutschland hat die Treiberentwicklung eingestellt. Die alten Treiber sind jetzt aber wieder verfügbar und: Es gibt neue! Diese liegen zwar nicht in Deutsch, sondern nur in Englisch vor, aber das dürfte kaum stö-

Newsarchive und Homepages

Name	*Adresse*	*Hinweise*	*Wertung*
buntspecht.de	http://www.buntspecht.de	Deutsches Newsarchiv. Sehr umfangreich, alle interessanten Themen rund um OS/2 gut abgedeckt.	●●●●
os2online	http://www.os2online.de	Online Magazin mit einem gut zusammengestellten Newsarchiv, ferner einer sehr empfehlenswerten Hardwareecke.	●●●
Warpcast	http://www.warpcast.com	Das englischsprachige Newsarchiv. Umfangreich, international und ein wenig besser aufgebaut als buntspecht.de	●●●●
OS/2 Warp Homepage	http://www.software.ibm.com/os/warp	Die OS/2 Warp Homepage von IBM. Allgemeine Produktinformationen, NetworkComputing etc. Auf dieser Seite stehen Java und die Netscape Produkte zum Download bereit.	●●●

ren. Die neuen Treiber unterstützen folgende Druckermodelle von EPSON:

- EPSON Stylus COLOR 600
- EPSON Stylus COLOR 800
- EPSON Stylus COLOR 850
- EPSON Stylus COLOR 1520
- EPSON Stylus COLOR 3000
- EPSON Stylus Photo
- EPSON Stylus Photo EX
- EPSON Stylus Photo 700

Das ist doch eine überaus erfreuliche Nachricht, da man wieder eine Alternative in Sachen Druckern hat! Erfreut durch die neuen Treiber werden wir uns die EPSON-Drucker einmal näher ansehen und sie mit Druckern anderer Hersteller, die ebenfalls eigene OS/2-Treiber bieten, vergleichen.

Die Treiber stehen zum Download auf dem EPSON FTP-Server bereit. Auf unserer Homepage finden Sie auf der Treiberseite entsprechende Links. Möchten Sie sich die Treiber direkt via FTP holen, finden Sie die nötige Adresse in unserem Verzeichnis *Treiber*.

Display Doctor für OS/2

SciTech entwickelt gerade eine OS/2-Version des *Display Doctors*, den es bekanntlich für DOS, Windows und Linux gibt. Das Release soll bald erfolgen. Die OS/2 Variante des *Display Doctors* wäre eine wahre Revolution für alle OS/2 Anwender: Der *Display Doc-*

Treiber

Name	***Adresse***	***Hinweise***	***Wertung***
Device Driver Pak Online	http://service.software.ibm.com/os2ddpak/index.html	Das OS/2 Device Driver Pak, wie man es von der CD zu Warp 4 kennt gibt es hier online. Die Seiten werden aktualisiert und enthalten viele neue Einträge. Ein guter Startpunkt für die Treibersuche im Netz.	●●●
The Notebook/2 Site	http://www.os2ss.com/users/DrMartinus/nbdriver.htm	Die Seite mit Treibern für OS/2-Notebooks auf der Homepage von »Dr. Martinus«. Treiber für Warp-taugliche Notebooks; viele Graphikkartentreiber und Unterstützung für neue Soundchips (z.B. ESS Maestro-2).	●●●●
EPSON Deutschland GmbH	ftp.epson.de/pub/de/driver/os2	Druckertreiber von EPSON. Die Treiber der EPSON Deutschland GmbH (Datei: epomni.exe) und die neuen englischen Treiber (Datei: ep1020.exe).	●●●●

tor ist ein Systemtool, mit dem man, gleich welche Graphikkarte man verwendet, Farbtiefe, Bildwiederholfrequenz usw. einstellen kann. *SciTech* entwickelt OS/2-Graphiktreiber, die DIVE und EnDIVE unterstützen und jede nur erdenkliche Auflösung bis 2045 x 1536 bieten, ferner alle Farbtiefen und Bildwiederholfrequenzen bis an die Grenzen, welche die Hardware vorgibt. Der *OS/2 Display Doctor* soll jeden Graphikchip unterstützen, angefangen bei ET3000 Graphikkarten für den ISA-Bus bis hin zu den neuesten Matrox Karten. Das würde heißen, man hätte keine Probleme mit dem leidigen Thema der mangelnden Graphikkartentreiber unter OS/2 mehr und im wahrsten Sinne des Wortes

OS/2-Benutzervereine (Deutschland)

Name	*Kontakt*
TeamOS/2 Berlin und OS/2 User Group Berlin	http://www.math.fu-berlin.de/~rene/os2/teamberlin.html
TeamOS/2 Böblingen	http://wwwcip.rus.uni-stuttgart.de/~phy27258/teambb.html
Stammtisch Hannover	j.wellhausen@teamos2.de
TeamOS/2 Weser Ems e.V. & Stammtisch Oldenburg	http://www.pmnet.uni-oldenburg.de/~teamos2
TeamOS/2 Franken	http://www.bwc.de/~teamos2
OS/2 User Group RheinMain	http://www.vey.de/os2ugrm
OS/2 Stammtsich OWL	silno@t-online.de
OS/2 Stammtisch Köln / TeamOS/2 Köln/Bonn e.V.	http://www.xept.com/teamos2
TeamOS/2 Region Trier	http://www.teamps2.ipcon.de
TeamOS/2 Region Hamburg e.V & OS/2 User Group Hamburg	http://www.teamos2.rightsource.de
TeamOS/2 Köln/Bonn e.V.	http://www.xept.com/teamos2
TeamOS/2 3-Länder-Eck (TeamOS/2-3LE)	http://privat.swol.de/UlfBartholomaeus/teamos2-3LE.html
Stammtisch Lörach	http://privat.swol.de/UlfBartholomaeus/Stammtische.html
Stammtisch Villingen-Schwenningen	http://privat.swol.de/UlfBartholomaeus/Stammtische.html
TeamOS/2 Ruhr e.V.	http://www.nordrhein.de/home/teamos2
Team OS/2 München Stammtisch	sterlike@ibm.net
Neuer OS/2-Dachverband	http://www.buntspecht.de/dachverband
TeamOS/2 Homepage (wird umgestaltet)	http://www.teamos2.de

die Qual der Wahl unter über 250 Graphikadaptern. Damit wird das Umsteigen von anderen Betriebssystem auf OS/2 auch wieder schmackhafter.
Eine Beta oder weitere Informationen sind auf der Homapage von *SciTech* noch nicht verfügbar. Allerdings existiert eine Mailingliste, in die man sich eintragen kann, um bezüglich des OS/2 *Display Doctors* auf dem neuesten Stand der Dinge gehalten zu werden. Ein Eintrag dürfte also nicht schaden. Die Homepage erreichen Sie unter *http://www.scitech.com*.
Bis zur nächsten Ausgabe werden wir versuchen, genauere Informationen von *Sci Tech* zu bekommen. Sobald der *Display Doctor* für OS/2 verfügbar ist, werden wir ihn sofort testen und Ihnen vorstellen. Mit solchen erfreulichen Meldungen klingt das Jahr aus. Seien wir also gespannt auf das nächste.

OS/2-Benutzervereine im Dezember
Nebenstehend finden Sie die wichtigsten Benutzervereine zu OS/2 mit einer entsprechenden Kontaktadresse im Netz. Diese Liste erhebt nicht den Anspruch auf Vollständigkeit. Wenn Sie selbst Mitglied einer OS/2-Benutzergruppe sind, auf ihre Gruppe aufmerksam machen und in unserer Liste geführt werden möchten, lassen Sie es uns wissen. Wie bei allen anderen Kontakten und Bezugspunkten werden neue Einträge stets in dieses Kapitel aufgenommen; alte im Anhang archiviert.
Die einzelnen User Groups, TeamOS/2-Vereine und OS/2-Stammtische freuen sich über neue Gesichter; jeder OS/2-Anwender ist also willkommen, auch wenn er vorher keinen Kontakt zum Verein hatte. Von elitären Grüppchen kann also kaum die Rede sein. Die Mitglieder der einzelnen Gruppen sind zudem technisch meist sehr versiert und stehen Ihnen daher für Fragen rund um OS/2 auch gerne zur Verfügung. Vielleicht ergibt sich ja für Sie die Möglichkeit zum Austausch mit einer der Gruppen.
Wir sind bemüht, Ihnen in den folgenden Ausgaben immer wieder einzelne Vereine und deren Arbeit vorzustellen. ❑

☎ Nummern, die man kennen sollte

In diesem Kasten finden Sie Telefonnummern und weitere Kontaktmöglichkeiten zu in diesem Band erwähnten Herstellern.

IBM
IBM Direkt 01805-313233
Vermittlung zum Ansprechpartner, allg. Informationen

OS/2 Line 0130-824457
Registrierung (kostenfrei)

OS/2 Line 069-66549050
Fragen zu OS/2; Warp 3 und Warp 4 (30 Tage nach der Registrierung kostenfrei)

Literaturservice 01805-5090
Techn. Literatur zu allen IBM Produkten.
Erreichbar: Mo.-Fr. 8.00-18.00 Uhr

Iomega
Iomega-BBS 02957-792990

Techn. Support 0130-829446 (Deutschland)
0660-5541 (Österreich)
0800-558091 (Schweiz, dt.)

Internet: http://www.iomega.com

Hardware Theorie 1 - Die CPU

Das Kernstück eines Rechners ist die Central Processing Unit (CPU). Die PCs, die wir heute einsetzen, basieren auf dem 80386-Prozessor von Intel, dessen Architektur OS/2 seit der Version 2.0 unterstützt. In dieser Ausgabe betrachten wir auch das Speichermanagement von OS/2 und werden uns daher zuvor mit dem 80386 auseinandersetzen, um zu verstehen, warum OS/2 wie mit dem Speicher umgeht.

Eine Betrachtung des 80386 genügt zum Verständnis, denn bis auf Verbesserungen hinsichtlich des Befehlssatzes und der Hinzufügung weiterer Baugruppen arbeiten die Folgemodelle, seien es die der Pentium-Familie von Intel oder kompatible Prozessoren der Firmen AMD oder Cyrix nach genau den gleichen Prinzipien, und OS/2 verhält sich mit diesen Prozessoren nicht anders als mit dem 80386.

Überblick zum 80386

Der 80386 ist der erste 32-bit Prozessor der Firma Intel gewesen. Als 32-Bit-Baustein kann er 32 **Bits** gleichzeitig verarbeiten, und verfügt über einen 32-bit breiten **Daten- und Adreßbus**. Da der Prozessor den Speicher mit 32 Adreßpfaden anspricht, kann er maximal 2^{32} Bit adressieren, also einen 4 Gbyte großen physikalischen Adreßraum zur Verfügung stellen. Virtuelle Adressen bildet der 80386 mit 48 Bit und verfügt daher über einen virtuellen **Adreßraum** von 64 TByte Größe.

Der 80386 ist abwärtskompatibel zu seinen Vorgängern 8086, 80186 und 80286, wodurch Programme, die für diese Prozessoren geschrieben wurden, ohne Veränderungen durch den 80386 ausgeführt werden können. Diese Abwärtskompatibiltät bieten auch alle Nachfolger des 80386, die wir später kurz betrachten werden.

Wie der 80286 kann der 80386 in zwei Betriebsmodi laufen, im **Real Mode** und im **Protected Mode**. Im *Real Mode* ist der Prozessor nichts anderes als ein schneller 8086. Im *Protected Mode* jedoch bietet der Prozessor folgende Leistungsmerkmale:

- Virtuelle Adressierung
- Paging
- Multitasking-Mechanismen
- Umfangreiche Schutzkonzepte
- den virtuellen 8086-Modus

Der 80386 bietet ein **segmentiertes Speichermodell**, genau wie seine Vorgänger. Damit ist er nicht nur hinsichtlich des Befehlssatzes, sondern auch des Speichermanagements abwärtskompatibel. Der einzige Unterschied hierbei liegt in der Größe der einzelnen Segmente, die 4 GByte betragen kann (im Gegensatz zu den 64 KByte-Segmenten des 80286). OS/2 1.x machte vom *segmentierten Speichermodell* des 80286 Gebrauch; ab der Version 2.0, die für den 80386 entwickelt wurde, änderte sich das, indem OS/2 ein **lineares Speichermodell** auf dem 80386 zur Verfügung stellt, das man auch **Flat Memory Model** nennt. Der Artikel *OS/2-Speichermanagement* in dieser Ausgabe zeigt, wie das lineare

Speichermodell realisiert wird. In diesem Kapitel betrachten wir lediglich den Prozessor und damit das Modell eines segmentierten Speichers. Das ist die Voraussetzung, um zu verstehen, wie andere Speichermodelle auf dem 80386 eingerichtet werden können.

Real- und Protected Mode

Wenn der Computer eingeschaltet oder ein Reset durchgeführt wird, arbeitet der 80386 im *Real Mode* und verhält sich wie ein 8086. Diese Betriebsart heißt deswegen *Real Mode*, weil sich alle Adressen, die der Prozessor verarbeitet, auf tatsächliche physikalische Adressen beziehen. In diesem Falle enthalten die Segmentregister (s.u.) die Basisadresse der einzelnen Speichersegmente, also DS das Datensegment, CS das Codesegment, SS das Stacksegment usw. Mit einem 16-Bit Offset wird dann auf einen Eintrag innerhalb des Segmentes zugegriffen. Damit kann ein **Segment** eine maximale Länge von 2^{16} Bit also 64 KByte annehmen. Im *Real Mode* werden nur 20 Adreßpfade benutzt, um den Speicher anzusprechen, d.h. der maximal andressierbare Speicher beläuft sich auf eine Größe von 1 MByte.

Die CPU kann jedoch durch die Betriebssystemsoftware in den *Protected Mode* umgeschaltet werden. Der physikalische Adreßraum steigt dabei auf 2^{32} Bit also 4 Gbyte, und verwendete Adressen verweisen nicht auf tatsächliche Speicherbereiche, sondern stellen nur Bezeichnungen für Speicherstellen im virtuellen Adreßraum dar, die von der CPU erst in physikalische Adressen übersetzt werden müssen. Mit anderen Worten: Im *Protected Mode* benutzen Programme ausschließlich virtuelle Adressen, was bedeutet, daß die Anwendungen keinen Zugriff auf aktuelle physikalische Adressen haben. Jeder Prozeß erhält im *Protected Mode* seinen eigenen virtuellen Adreßraum, der theoretisch eine Größe von bis zu 64 TByte erreichen kann, und vermag damit nicht auf die Adressen anderer Prozesse zuzugreifen. Wegen dieses Schutzprinzips, das durch die virtuelle Adressierung hervorgeht, erhält der zweite Betriebsmodus des 80386 auch seinen Namen *Protected Mode*.

Da OS/2 ein Protected Mode System ist, werden wir uns im folgenden detailliert mit den Leistungsmerkmalen des 80386 im *Protected Mode* befassen. Zuvor wollen wir aber den internen Aufbau des Prozessors genauer betrachten.

Einheiten der CPU

Der 80386 verfügt über mehrere verschiedene Einheiten, die alle über einen 32-bit-Bus miteinander verbunden sind. Abb. 1 zeigt den Aufbau des Prozessors. Wie man sieht, stehen für verschiedene Aufgaben in der CPU sechs Einheiten zur Verfügung, die basierend auf dem Prinzip des **Pipelining** zusammenarbei - ten.

Die *Bus Interface Unit (BIU)* dient als Schnittstelle der CPU zur Außenwelt des Rechners. Jede andere der 80386-Einheiten wickelt Ein- und Ausgabeoperationen über die *BIU* ab, welche entsprechende Zugriffe auf die weitere Hardware des Systems durchführt. Die *BIU* arbeitet mit physikalischen Adressen, d.h. logische Adressen müssen erst von der *Segmenta-*

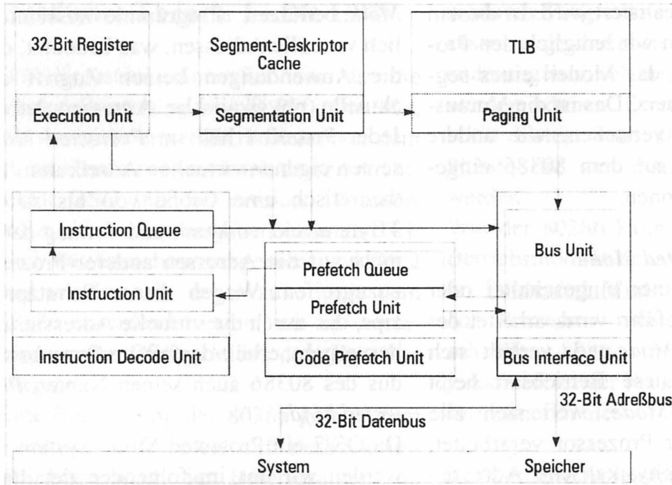


Abb. 1: Die Architektur des 80386

von 16 Bytes haben kann). Ist ein Befehl vollständig aus der Warteschlange eingelesen, überträgt die *IDU* den Befehl in ein maschineninternes Format. Der decodierte Befehl wird in der Befehlswarteschlange der *IDU* abgelegt, welche insgesamt drei Einträge enthalten kann.

tion und Paging Unit in physikalische Adressen umgewandelt werden, ehe sie an die *BIU* übergeben werden

Die *Instruction Prefetch Unit (IPU)* verwaltet eine Warteschlange, die sie ständig mit Befehlen gefüllt zu halten versucht. Dazu fordert sie von der *BIU* stets die Adresse des nächsten Befehls im Speicher an und legt den entsprechenden Befehl in der Warteschlange ab. Anfragen der *IPU* an *BIU* werden mit einer geringeren Priorität behandelt als Anfragen anderer Einheiten, so daß die Ausführung gerade bearbeiteter Befehle nicht verlangsamt wird. Trotzdem liest die *IPU* so oft wie möglich neue Befehle aus dem Speicher ein, um sie in die Warteschlange zu legen.

Die *Instruction Decode Unit (IDU)* liest einzelne Bytes aus der *IPU* und bestimmt, wieviel Bytes notwendig sind, um einen Befehl zu vervollständigen (wobei ein einzelner Befehl eine Länge

Die *Execution Unit (EU)* führt die eigentliche Rechenarbeit in der CPU durch (Multiplikationen, Divisionen, Bitverschiebungen usw.). Die einzelnen Register des 80386 sind ebenfalls in dieser Einheit enthalten. Für Ein- und Ausgabeoperationen gibt die *EU* der *BIU* ein entsprechendes Signal, um auf die übrige Hardware des Systems zugreifen zu können.

Die *Segmentation Unit (SU)* übersetzt Segmentadressen in linerare Adressen. Diese Einheit verwaltet auch einen Pufferspeicher, in dem Tabellen zur Speicherverwaltung (sogenannte Deskrtiptortabellen) angelegt werden. Werden die Pagingfähigkeiten des 80386 nicht genutzt, sind die linearen Adressen gleichzeitig physikalische Adressen, die für Speicherzugriffe an die *BIU* übergeben werden können. Ist das Paging des Prozessors aktiviert, wird der lineare Adreßraum in 4096 Byte große Blöcke

unterteilt, die man als **Pages** (Seiten) bezeichnet. Jede dieser Seiten kann dabei auf eine ganz andere physikalische Stelle im Speicher abgebildet werden.

Die *Paging Unit (PU)* übersetzt die von der *SU* erzeugten linearen Adressen in physikalische Adressen, sofern das Paging aktiviert ist. Die *PU* benutzt zur Übersetzung entsprechende Tabellen, um die einzelnen Seiten den physikalischen Speicherstellen zuweisen zu können. Daneben verfügt die *PU* über einen Pufferspeicher, den sogenannten **TLB (Translation Lookaside Buffer)**, der den Zeitaufwand für eine seitenorientierte Speicherverwaltung erheblich verringert.

Die Register des 80386

Der 80386 enthält insgesamt 34 **Register**. Sechzehn davon können vom Anwendungsentwickler verwendet werden, die restlichen 18 stehen nur dem Betriebssystem zur Verfügung. Die einzelnen Register der beiden Gruppen zeigt Abb. 2. Sie haben folgende Bezeichnungen und Aufgaben:

1. Allgemein verfügbare Register

1.1. **Acht** *32-bit allgemeine Register* namens EAX, EBX, ECX, EBP, ESP, ESI und EDI, welche Operanden für logische und arithmetische Operationen aufnehmen. Die niederwertigen **Worte** der 32-Bit-Register haben eigene Namen (AX, BX, CX, DX, BP, SI, DI, IP) und können als individuelle Einheiten behandelt werden, wodurch der 80386 mit seinen 16-Bit Vorgängern kompatibel ist. Jedes 16-Bit-Register AX, BX, CX, DX ist unterteilt in ein Highbyte (AH, BH, CH, DH) und ein Lowbyte (AL, BL, CL, DL). Damit ist auch der Umgang mit 8-Bit Programmen und Daten gewährleistet.

1.2. **Sechs** Segment-Register mit der Bezeichnung CS (Codesegment) SS (Stacksegment), DS, ES, FS und GS (Datensegmente). Diese

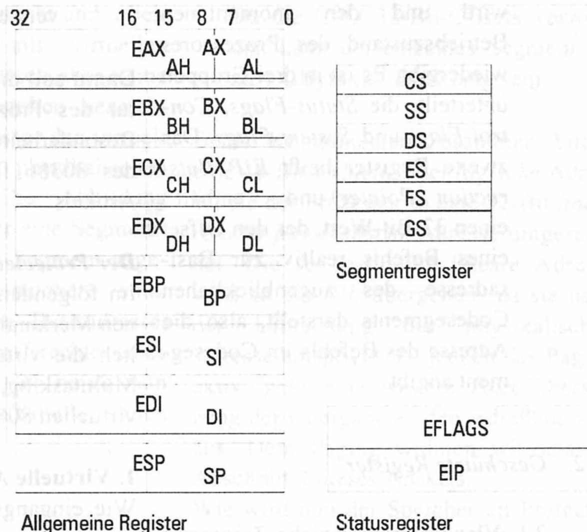


Abb. 2: Die für Anwendungsprogramme zugänglichen Register des 80386 (Generelle Register)

Register identifizieren die Speichersegmente, die für die Ausführung eines Programms zur Verfügung stehen. Das Segment, das die momentan auszuführenden Befehle eines Programmoduls enthält, wird durch das Segment-Register CS spezifiziert. Um Unterprogramme einer Anwendung aufrufen zu können, ist eine Speicherregion erforderlich, die als **Stack** bezeichnet und durch das Stack-Register SS spezifiziert wird. Die Datensegmentregister enthalten die Spezifikationen der unterschiedlichen Datensegmente, die für das augenblicklich laufende Programm adressierbar sind.

1.3. **Zwei** Status- und Instruction-Register: Das 32-Bit Flag Register, das als *EFLAG* bezeichnet wird und den momentanen Betriebszustand des Prozessores wiedergibt. Es ist in drei Gruppen unterteilt, die *Status-Flags*, *Control-Flags* und *System-Flags*. Das zweite Register heißt *EIP (Instruction Pointer)* und enthält einen 32-Bit-Wert, der den Offset eines Befehls realtiv zur Basisadresse des augenblicklichen Codesegments darstellt, also die Adresse des Befehls im Codesegment angibt.

2. *Geschützte Register*

2.1. **Vier** Register, welche Zeiger auf Datenstrukturen aufnehmen, die notwendig sind, um das segmentierte Speichermodell zu implementieren: **GDTR** (Global Descriptor Table Register), **LDTR** (Local Descriptor Table Register), **IDTR** (Interrupt Descriptor Table Register und **TR** (Task Register).

2.2. Vier Register, die Zeiger auf Datenstrukturen aufnehmen, welche notwendig sind, um das Paging zu implementieren und Statusinformationen zu speichern: CR0 bis CR3.

2.3. Acht Debugregister, um das Debugging von Echtzeitsystemen und Applikationen zu unterstützen: DR0 bis DR7.

2.4. Zwei Testregister, die zur Prüfung der Integrität des TLB (Translation Lookaside Buffers) verwendet werden.

Damit soll die Betrachtung der Architektur des Prozessors abgeschlossen sein. Besonderheiten der Nachfolgemodelle des 80386 finden Sie am Ende dieses Artikels.

Der Protected Mode

Im folgenden bertrachten wir die einzelnen Merkmale des *Protected Mode*, nämlich die virtuelle Adressierung, Paging, Multitasking, Schutzkonzepte und den virtuellen 8086 Modus.

1. Virtuelle Adressierung

Wie eingangs erwähnt, stellt der 80386 ein segmentiertes Speichermodell zur Verfügung, d.h. der Speicher wird in Segmente unterteilt. Jeder Prozeß erzeugt

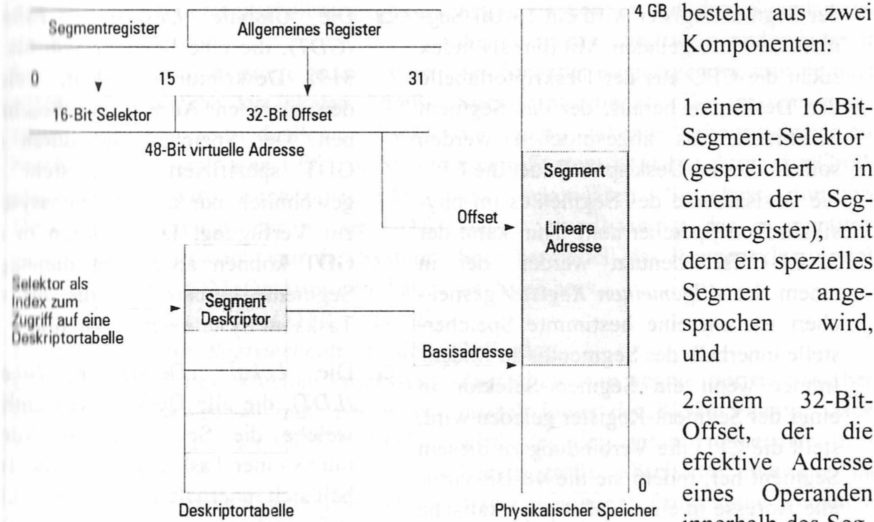


Abb. 3: Die virtuelle Adressierung des 80386. Der physikalische Speicher wird über Deskriptoren abgebildet. Programme arbeiten nur mit virtuellen Adressen.

dabei seine eigenen Segmente und arbeitet im *Protected Mode* mit virtuellen Adressen. Da diese Adressen aber keine physikalischen Speicherstellen bezeichnen (s.o.), muß der Prozessor den virtuellen Adreßraum auf den physikalischen Adreßraum abbilden. Die Abbildung erfolgt segmentweise über eine Segment-Tabelle, in der für jedes Segment sogenannte **Deskriptoren** gespeichert sind, welche die physikalische Basisadresse, die Länge und die Zugriffsrechte eines Segmentes enthalten. Daher nennt man eine solche Segment-Tabelle auch Deskriptor-Tabelle.

Virtuelle und physikalische Adressen

Eine virtuelle Adresse, wie sie von einem Prozeß im System verwendet wird, besteht aus zwei Komponenten:

1.einem 16-Bit-Segment-Selektor (gespreichert in einem der Segmentregister), mit dem ein spezielles Segment angesprochen wird, und

2.einem 32-Bit-Offset, der die effektive Adresse eines Operanden innerhalb des Segmentes angibt, also seine Entfernung vom Anfang des Segmentes.

Da die CPU 32-Bit-Offsets verwendet, kann ein einzelnes Segment bis zu 4 GByte (2^{32} Bits) lang sein.

Beide Komponenten zusammen bilden eine 48-Bit *virtuelle* oder *logische* Adresse, die der Prozessor in eine 32-Bit *lineare* oder *physikalische* Adresse umgerechnet. Die so bestimmte lineare Adresse wird an die *BIU* übergeben, da sie nach der Umrechung der physikalischen Adresse entspricht. Ist jedoch das Paging aktiv, geht eine weitere Adreßübersetzung der Übergabe an den Adreßbus voraus. Dem Paging widmen wir uns im Abschnitt 2 dieses Artikels.

Wie wird nun der Speicher im Protected Mode adressiert? Abbildung 3 zeigt den Vorgang der Übersetzung einer virtuellen in eine physikalische Adresse: Aus einem

der Segmentregister wird ein 16-Bit-Segment-Selektor geladen. Mit ihm als Index sucht die CPU aus der Deskriptortabelle den Deskriptor heraus, der das Segment beschreibt, das angesprochen werden soll. In diesem Deskriptor findet die CPU die Basisadresse des Segmentes im physikalischen Speicherraum. Nun kann der 32-Bit-Offset benutzt werden, der in einem der *Allgemeinen Register* gespeichert ist, um eine bestimmte Speicherstelle innerhalb des Segmentes zu finden. Immer wenn ein Segment-Selektor in eines der Segment-Register geladen wird, stellt die CPU die Verbindung zu diesem Segment her, indem sie die 48-Bit-virtuelle Adresse in eine 32-Bit-physikalische Adresse umrechnet. Sind die Segment-Register einmal initialisiert, ist für den Zugriff auf einen Operanden in einem der Segmente nur noch ein Near-Pointer, also der 32-Bit-Offset erforderlich. Nur wenn ein Segment neu geladen wird, aktiviert die CPU den Mechanismus für die Adressübersetzung.

Deskriptoren und ihre Tabellen
Wie man sicher schon vermutet hat, sind die Deskriptoren der Dreh- und Angelpunkt des gesamten Speichermanagements und -wie wir später sehen werden- auch der Schutzkonzepte. Sie werden in den bereits erwähnten Desktriptor-Tabellen eingetragen. Eine Deskriptor-Tabelle ist eine vom System erzeugte, speicherresidente Tabelle. Innerhalb eines *Protected Mode*-Systems existieren mehrere *Deskriptor-Tabellen*. Für die virtuelle Adressierung interessant sind:

- Die *Globale Deskriptor Tabelle (GDT)*, die eine Sequenz von bis zu 8192 Deskriptoren enthält, welche den globalen Adressraum beschreiben. Der Speicher, der durch die GDT spezifiziert wird, steht für gewöhnlich nur dem Betriebssystem zur Verfügung; Deskriptoren in der GDT können aber auch diejenigen Segmente beschreiben, die für alle Tasks im System verfügbar sind.
- Die *Lokale Deskriptor Tabelle (LDT)*, die alle Deskriptoren enthält, welche die Segmente des Adreßraums einer Task angeben. Jede Task hält sich innerhalb ihrer eigenen LDT auf, wodurch der Zugriff auf Segmente anderer Tasks ausgeschlossen ist; damit werden die physikalischen Segmente unterschiedlicher Tasks voneinander getrennt.

Der Vollständigkeit halber sei noch die dritte Deskriptortabelle erwähnt, die in einem Protected Mode-System existiert, die *Interrupt Deskriptor Tabelle (IDT)*, welche Deskriptoren enthält, die auf die Basisadressen von Unterbrechungsroutinen (Interrupt-Handler) verweisen.

Die GDT und die IDT sind in einem *Protected Mode*-System nur einmal vorhanden, weshalb sie nicht durch Deskriptoren beschrieben werden, sondern durch die 48-Bit-Register *GDTR (Global Descriptor Table Register)* und *IDTR (Interrupt Descriptor Table Register)*, welche die physikalischen Basisadressen und die Tabellenobergrenzen der beiden Tabellen festlegen. Beide Register wer-

den zu Beginn der Initialisierungsphase des Systems geladen.
Für jeden Prozeß existiert jedoch eine eigene LDT, welche den lokalen Adreßraum eines individuellen Tasks beschreibt. Wo sich eine LDT im Speicher befindet, wird durch einen LDT-Deskriptor festgelegt, der in der GDT gespeichert wird (womit nur das System Zugriff auf die LDT-Deskriptoren hat). Das 48-Bit-Register *LDTR (Local Descriptor Table Register)* enthält den LDT-Deskriptor des aktuellen Prozesses. Bei einem Taskwechsel muß ein neuer LDT-Deskriptor aus der GDT in LDTR geladen werden.
Wer am Aufbau eines Segment-Selektors und der Segment-Deskriptoren interessiert ist, schlage bitte im Lexikon unter **Selektor** nach.

Durch das segmentierte Speichermodell werden die Adreßräume sowohl des Systems als auch aller anderen Prozesse im System voneinander getrennt, je nachdem zu welchen Prozessen die Segmente gehören. Diese werden durch die LDTs den einzelnen **Tasks** zugeordnet. Dadurch wird eine sehr sichere Betriebsumgebung zur Verfügung gestellt, die hardwareseitig implementiert ist.
Daneben bietet die Verwendung virtueller Adressen für das einzelne Programm den Vorteil, nicht mehr mit physikalischen Speicherreferenzen arbeiten zu müssen. Wie die Segmente im Arbeitsspeicher des Systems abgelegt werden, liegt in der Hand des Betriebssystems. Ein Programm sieht nur seinen virtuellen Adreßraum. Segmente können aus dem Speicher auf die Festplatte übertragen und wieder in den Speicher eingelagert werden, je nach Speicherplatzbedarf. Der maximale zur Verfügung stehende Arbeitsspeicher auf einem System ist daher durch die Größe des sekundären Speichers limitiert. Damit vervollständigt das segmentierte Speichermodell das Konzept des virtuellen Speichers.

2. Paging

Zusätzlich zum segmentierten Speichermodell bietet der 80386 Paging. Dabei wird der der gesamte Speicher in 4 KByte große Abschnitte unterteilt, die man Pages nennt. Programme können so in mehrere Abschnitte gleicher Länge unterteilt werden, ohne daß aber diese Abschnitte eine Beziehung zur logischen Struktur des Programms haben. Der Vorteil der Pages liegt darin, daß sich Pages nur so lange im Speicher aufhalten, wie sie für die Ausführung des erforderlichen Programmabschnittes benötigt werden. So ist man auch nicht gezwungen, Segmente in einem Stück aus dem Arbeitsspeicher auf die Festplatte zu übertragen, was besonders bei großen Segmenten zu Performanceverlusten führen würde, sondern in handlichen 4 KByte Stücken. Die ausgelagerten Pages finden sich dann in sogenannten Swap-Dateien. Sie bestehen aus einer Ansammlung 4 KByte großer Speicherblöcke. Ist der Paging-Mechanismus aktiv, übersetzt der 80386 eine virtuelle Adresse in zwei Schritten in eine physikalische Adresse:

1. Mit dem Segment-Übersetzungs-Mechanismus berechnet die CPU die

lineare Adresse aus einem Segment-Selektor und einem Segment-Offset. Die resultierende lineare Adresse ist mit der 32-Bit physikalischen Adresse identisch. Dieser Mechanismus ist im Protected-Mode *immer* aktiv.

2. Der zweite Schritt, die Benutzung des Paging-Übersetzungs-Mechanismus, ist optional. Das Paging ist nur dann aktiv, wenn das PG-Bit (Paging Enable) im CR0-Register des Prozessors gesetzt ist (PG=1). Dann wird der Paging-Übersetzungs-Mechanismus dem Segment-Übersetzungs-Mechanismus nachgeschaltet.

Das Paging ist ein zweistufiger Übersetzungsmechanismus. Für die weitere Verarbeitung linearer Adressen werden zwei Tabellentypen benutzt, welche zur Verwaltung der einzelnen Pages vom System angelegt werden:

1. die **Page-Directory-Tabelle**, welche die Position der Page-Tables im Speicher angibt, und

2. die **Page-Tables**, welche die Positionen der Pages im Speicher beinhaltet.

Beide Tabellentypen enthalten 32-Bit-Einträge und sind speicherresident. Wie die Anwenderpages haben sie eine invariable Länge von 4KByte. Die Position der Page-Directory-Tabelle im Speicher legt das 80386-Control-Register CR3 fest. Bevor der Paging-Mechanismus aktiviert werden kann, muß das CR3-Register initialisiert sein.

Beim Paging bezieht sich die lineare Adresse eines Operanden, die der Segment-Übersetzungs-Mechanismus liefert, nur noch indirekt auf dessen physikalische Adresse. Das ist darin begründet, daß die lineare Adresse in drei Felder unterteilt ist, die dem Zugriff auf die beiden Tabellen dienen. Diese drei Felder sind:

1. das *DIR-Feld* von Bit 22-31 der linearen Adresse, das als Index zur Auswahl von Einträgen in der Page-Directory-Tabelle dient,

2. das *TABLE-Feld* von Bit 12-21 der linearen Adresse, das als Index zur Auswahl von Einträgen in einer Page-Table liefert,

3. das *OFFSET-Feld* von Bit 0-11 mit dem ein individueller 32-Bit Eintrag in einer Anwender Page selektiert wird.

Wie eine lineare Adresse über das Paging in eine physikalische Adresse umgewandelt wird, zeigt Abbildung 4: Die Basisadresse der Page-Directory-Tabelle befindet sich im Controlregister CR3. Mit Hilfe des DIR-Feldes der linearen Adresse als logischen Offset kann dann ein bestimmter Eintrag in der Page-Directory-Tabelle selektiert werden. Jede Eintragung in der Page-Directory-Tabelle ist ein **Page-Table-Deskriptor**, der die physikalische Basisadresse einer Page-Table und Informationen über sie enthält. Die Page-Directory-Tabelle kann bis zu 1024 solcher Deskriptoren aufnehmen, also können bis zu 1024 unterschiedliche Page-Tables selektiert werden. Da das DIR-Feld 10 Bit lang ist, kann es maximal 2^{10} = 1024 Indexnummern liefern

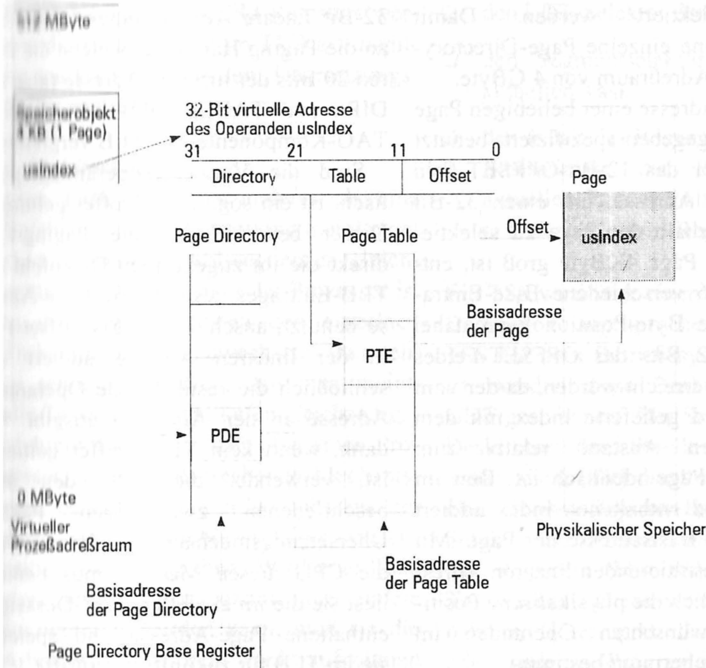


Abb. 4: Paging. Der Speicher wird in 4 KByte große Seiten (Pages) unterteilt, die komfortabel verwaltet werden können.

Nach der Spezifizierung der Basis-Adresse einer beliebigen Page-Table, benutzt der Prozessor das 10-Bit-TABLE-Feld der linearen Adresse, um einen bestimmten Eintrag in der Page-Table zu selektieren. Auch die Page-Table kann bis zu 1024 Eintragungen enthalten, die sogenannten **Page-Deskriptoren,** die das gleiche

Weil jeder Eintrag in der Page-Directory-Tabelle vier Bytes groß ist, muß die CPU die vom DIR-Feld vorgegebene Indexnummer mit 4 multiplizieren, um einen bestimmten Eintrag im Page-Directory auch physikalisch zu erreichen. Das Resultat dieser Operation ist dann identisch mit dem physikalischen Abstand eines 32-Bit-Eintrages im Page-Directory realtiv zu dessen Anfang. Diesen Offset-Wert addiert die CPU nun zur Basisadresse der Page-Directory-Tabelle, die sich im CR3-Register befindet. Die daraus resultierende lineare Adresse entspricht der physikalischen Position der gewünschten Page-Table.

Format wie die Page-Table-Deskriptoren haben. Die Page-Deskriptoren liefern die physikalische Basisadresse und Informationen einer einzelnen Page. Das Schema, mit dem die CPU die Basisadresse einer Page im Speicher findet, ist identisch mit dem, welches zur Berechung der Basisadresse einer Page-Table dient, so daß es hier nicht noch einmal wiederholt werden muß. Da eine einzelne Page-Table 1024 verschiedene Pages zu je 4 KByte adressieren kann, beschreibt sie einen Adreßraum von 4 MByte. Stehen in der Page-Directory-Tabelle alle 1024 Page-Table-Deskriptoren zur Verfügung, können auch 1024 verschiedene Page-

Tables selektiert werden. Damit beschreibt eine einzelne Page-Directory-Table einen Adreßraum von 4 GByte.
Ist die Basisadresse einer beliebigen Page wie oben angegeben spezifiziert, benutzt der Prozessor das 12-Bit-OFFSET-Feld der linearen Adresse, um einen 32-Bit Eintrag innerhalb der Page zu selektieren. Da jede Page 4KByte groß ist, enthält sie 4096 verschiedene Byte-Eintragungen. Jede Byte-Position kann dabei durch die 12 Bits des OFFSET-Feldes physikalisch erreicht werden, da der vom OFFSET-Feld gelieferte Index mit dem physikalischen Abstand relativ zum Beginn der Page identisch ist. Den im OFFSET-Feld enthaltenen Index addiert die CPU zur Basisadresse der Page. Mit der daraus resultierenden linearen Adresse ist schließlich die physikalische Position des gewünschten Operanden im 4GByte-Speicherraum bestimmt.

Der TLB

Da wir gesehen haben, wie komplex der Paging-Mechanismus arbeitet, würde er die Leistungsfähigkeit des Prozessors erheblich vermindern, wenn bei jeder Speicherreferenz auf beide Tabellen (Page Directory und Page Table) zugegriffen werden müßte. Daher verfügt der 80386 über ein für den Anwender transparentes Speicherblock-System (Cache), das als **Translation Lookaside Buffer (TLB)** bezeichnet wird und zum Zweck des schnellen Operandenzugriffs diejenigen Page-Frame-Adressen aufbewahrt, die vom Programm in jüngster Zeit benutzt worden sind.
Der TLB arbeitet wie folgt: Die Segmentierungs-Einheit des Prozessors liefert die 32-Bit lineare Adresse eines Operanden an die Paging Hardware, welche die oberen 20 Bits der linearen Adresse (also das DIR- und TABLE-Feld) mit allen 32 TAG-Komponenten im TLB vergleicht.
Sind die Vergleichsoperanden identisch, ist ein sog. TLB-Treffer gelungen. Dieser bewirkt, daß die Paging-Unit direkt die im zugehörigen Datenfeld des TLB-Eintrages gespeicherte Page-Adresse benutzt, anschließend das Offset-Feld in der linearen Adresse addiert und schließlich die resultierende Operanden-Adresse an den Adreßbus ausgibt. Nur dann, wenn kein TLB-Treffer gelungen ist, verwendet die CPU den oben beschriebenen zweistufigen Paging-Übersetzungsmechanismus. Immer wenn die CPU diesen Mechanismus benutzt, liest sie die im aktuellen Page-Deskriptor enthaltene Page-Adresse und speichert sie im TLB für zukünftige Zugriffe.
Statistiken zeigen, daß der TLB in vielen Multitaskingsystemen eine Trefferrate von bis zu 98% erreicht, womit die CPU nur bei etwa 2% aller Speicherreferenzen auf den Paging-Übersetzungsmechanismus zugreifen muß. Das Betriebssytem muß beim Paging:

- die Page-Tabellen initialisieren und Page-Fehler behandeln und
- den TLB als ungültig markieren, wenn irgend eine Änderung in einer der Page-Table-Eintragungen vorgenommen worden ist, weil in diesem Falle die in einem TLB-Eintrag gespeicherten und zusammengehörigen TAG- und DATEN-Informationen nicht mehr der realen Situation entsprechen.

Durch Neuladen des CR3-Registers werden die Eintragungen im TLB als ungültig markiert und aus dem Übersetzungspuffer »entfernt«.

I. Multitasking

Eine **Task** ist nichts anderes als eine Befehlssequenz. Multitasking meint nicht etwa, mehrere Befehlssequenzen gleichzeitig auszuführen; aber der Prozessor ist in der Lage, zwischen den einzelnen Tasks hin- und herzuwechseln. Das geht so schnell vonstatten, daß man meint, alle Tasks im System liefen gleichzeitig.
Mehrere Tasks benutzen also einen Prozessor, so daß entsprechende Mechanismen bereitgestellt werden müssen, um die Ausführung mehrerer Tasks »gleichzeitig« zu ermöglichen. Wird eine Task etwa unterbrochen, damit eine andere auszugeführt werden kann, muß sie die Arbeit wieder an dem Punkt aufnehmen können, an welchem ihr die Kontrolle über den Prozessor entzogen wurde.

Implemetierung auf dem 80386

Der 80386 unterstützt zu diesem Zweck ein spezielles Segment, das **Task-Status-Segment**, kurz TSS. In diesem Segment werden alle taskspezifischen Informationen gespeichert, so daß es der CPU möglich ist, zu allen Tasks umzuschalten und dabei zu gewährleisten, daß alle Tasks voneinander getrennt sind.
Jede Task im System bekommt ihr eigenes TSS. Die wichtigsten Informationen darin sind:

- Die Werte aller für das Programm zugänglichen Register (Allgemeine Register),
- den LDT-Selektor der Task,
- den Segmentselektor des TSS der aktuellen Task,
- den Inhalt des Registers CR3, das die Basisadresse der Page-Table der aktuellen Task enthält, und die Inhalte anderer Control-Register.

Ein TSS wird über einen speziellen TSS-Deskriptor angesprochen. Weil jedes TSS nur dem Betriebssystem zugänglich ist, werden diese Deskriptoren in der GDT gespeichert. Der Selektor des TSS-Deskriptors der aktuellen Task wird im Taskregister (TR) des Prozessors gespeichert. Dieses Register bestimmt nur den Selektor der aktiven Task; ein Neuladen des Registers führt jedoch zu keinem Taskwechsel. Diese werden durch Ausnahmen (Interrupts), der aktiven Task, oder anderen Programmen ausgeführt, die für den Taskwechsel bestimmt sind.

Was bei Taskwechseln passiert

Zunächst werden die Inhalte der Register im TSS der aktuellen Task gesichert, damit bei einem erneuten Taskwechsel diese Task ihre Arbeit korrekt fortführen kann. Dann lädt das Betriebssystem einen neuen TSS-Selektor in das Taskregister, sucht mit dessen Hilfe den entsprechenden Deskriptor aus der GDT und bestimmt die Adresse des TSS im Arbeitsspeicher. Nachdem das TSS der neuen Task gefunden wurde, werden die Register mit den entsprechenden Werten im TSS der neuen Task geladen. Nachdem der LDT-Selektor aus dem TSS in das LDTR geladen worden ist, kann der entsprechende LDT-Deskriptor aus der

GDT gesucht und auf die LDT der Task zugegriffen werden, um die Segmentregister zu initialisieren. Anschließend kann die neue Task zur Ausführung gelangen.

Dieser TSS-Mechanismus wird vom Betriebssystem verwaltet. Es muß ferner für jede neue Task ein neues TSS bereitstellen und entsprechend initialisieren. Ist für jede Task im System ein TSS verfügbar, kann zwischen den einzelnen Tasks umgeschaltet werden. OS/2 macht davon aber wenig Gebrauch, sondern stellt eine eigene Taskverwaltung zur Verfügung. Da wir in diesem Artikel nur die Fähigkeiten auf Prozessorebene betrachten, soll eine genaue Erklärung des OS/2-Multitasking-Mechanismus einer der nächsten Ausgaben vorbehalten sein.

4. Systemschutz

Die Schutzhardware des 80386 stellt den Umgang mit dem Speicher, allen I/O-Ressourcen und den Befehlssätzen unter Regeln. Diese Schutzkonzepte greifen tief in die Speicherverwaltung ein, da sie eng mit der Memory Management-Hardware verbunden sind. Der Systemschutz hat zum Zweck, bestimmte Operationen auf eine gewissen Teilmenge bestimmter Tasks einzuschränken. Jede Task, welche gegen diese Schutzkonzepte verstößt, wird von der Schutzhardware festgestellt, als Schutzverletzung signalisiert und abgebrochen.

Durch die Schutzhardware kann die Betriebssystemsoftware von anderen Programmen isoliert werden, womit das Betriebssystem ungestört die Ressourcen eines Systems verwalten kann, eine Notwendigkeit für Multitasking-Systeme wie OS/2. Auch die einzelnen Anwendunge können voneinander getrennt werden. S können Fehler, die in ein einem Pro gramm auftreten, das Verhalten eine anderen nicht beeinflussen, wodurch di Zuverlässigkeit der Anwendungen un die Systemstabilität steigt.

Der 80386 bietet zum Systemschutz in wesentlichen die Überprüfung vo Datentypen, die Limitüberprüfung un unterschiedliche Privilegebenen, au denen die einzelnen Anwendungen lau fen.

Der Kernbaustein des 80386-Schutzme chanismus ist dabei der Segment Deskriptor. Wie wir wissen, sind i Deskriptoren Informationen über ei Segment enthalten, etwa die Basisadress eines Segmentes. Der Deskriptor beinhal tet aber auch Informationen für di Bereitstellung der Schutzkonzepte: De Typ eines Segmentes; die Größe des Seg mentes (Obergrenze, Limit); und desse Privilegebene. Auf diese Schutzdate bezieht sich die CPU immer zweimal nämlich beim Laden eines Segmentregi sters und bei jedem Zugriff auf Daten de sich gerade in Benutzung befindende Segments. Die Überwachung de Zugriffs auf Segmente ist daher Vorraus setzung für den Softwareschutz.

Die Überprüfung von Datentypen

Mit Hilfe des Segmenttyps sorgt die CP dafür, daß innerhalb von Programme alle Code- und Datensegmente richti benutzt werden. Beispielsweise könne Daten nicht ausgeführt und Code nich modifiziert werden; im CS-Register kan nur ein Selektor eines Codesegmente aufgenommen, solche Selektoren könne

jedoch nicht in eines der Datensegmentregister geladen werden. Datensegmente können ferner den Typ Read-Only oder Read/Write haben, womit nur Datensegmente letzteren Typs modifiziert werden können; für Codesegmente existieren die beiden Typen Execute-Only und Execute/Read, womit nur Codesegmente letzteren Typs gelesen werden können. Wenn das Paging aktiviert ist, wird eine zweite Ebene der Datentypüberprüfung durchgeführt, weil Pages vom Typ Read-Only oder Read/Write sein können[2].
Jede Operation, welche diese Schutzregeln verletzt, stellt eine **Exception** dar und führt zur Unterbrechung des Programms, das die Schutzverletzung ausführte. Diese bei jedem Speicherzugriff durchgeführten Überprüfungen gehören zum einfachen Typ des im 80386 implementierten Schutzmechanismus, der sicherstellt, daß mehrere Tasks voreinander geschützt sind.

Die Limitüberprüfung

Die im Limit-Feld des Segment-Deskriptor gespeicherte Obergrenze eines Segmentes benutzt der Prozessor zur sogenannten Limitüberprüfung, um ein Programm davon abzuhalten, über die Grenzen seines Segmentes hinaus Speicher zu adressieren. Während der Übersetzung von virtuellen in physikalische Adressen wird das Offset der virtuellen Adresse mit dem Wert im Limitfeld verglichen. Ist es größer als die Obergrenze des Segmentes, wird eine Exception ausgelöst, und das Programm, das den ungültigen Zugriff versuchte, wird beendet. Damit ist es einem Programm unmöglich, Daten in den Speicherbereichen anderer Programme zu modifizieren. Das ist übrigens mit ein Grund, warum es für Protected Mode-Systeme kaum Viren gibt, deren Prinzip es ist, die Speichereinträge anderer Programme zu manipulieren.

Die Privilegebenen

Um die Betriebssystemsoftware von den übrigen Anwendungen zu isolieren, ist eine weitere Dimension des Schutzes erforderlich, die sicher die bekannteste und auch die bedeutendste ist: Die **Privilegebenen**. Jedem Code- und Datensegment wird eine solche Privilegebene zugeordnet. Welcher Ebene ein Segment angehört, wird im Segment-Deskriptor und im Selektor festgelegt. Das RPL-Feld des Selektors im CS-Register gibt dabei an, welcher Privilegebene ein gerade ausgeführtes Programm angehört. Diese Privilegebene bezeichnet man als Current Privilege Level (CPL).
Der 80386 stellt eine Privileg-Hierarchie mit vier Ebenen zur Verfügung. Die Ebenen werden von innen nach außen von 0 beginnend durchnummeriert, wobei Ebene 0 die Ebene mit den meisten Privilegien ist, während Ebene 3 am wenigsten priviligiert und damit am ehesten Störmöglichkeiten anderer Programme ausgesetzt ist. Eine Darstellung dieses Ringsystems zeigt Abb. 5.
Auf Ebene 0 (CPL=0) befindet sich meist der Betriebssystemkern (Kernel), Deskriptor-Tabellen, Systemdaten usw; auf Ebene 1 (CPL=1) systemspezifische Erweiterungen und Service-Programme;

2 Nur der Inhalt von Pages mit dem Read/Write-Attribut kann modifiziert werden. Auf der Ebene der Pages existiert das Execute-Only-Attribut nicht mehr.

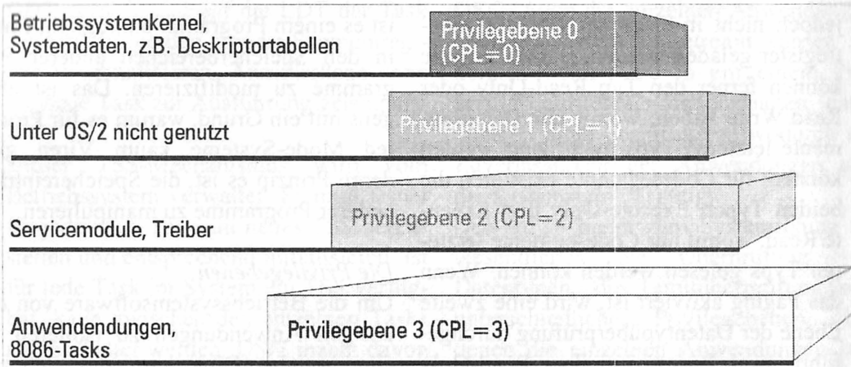


Abb. 5: Die vier Privilegebenen des 80386. Die Privilegebene eines Programmes wird im Selektor seines Codesegmentes gespeichert. Der

auf Ebene 2 (CPL=2) halten sich Programme auf, die nicht zum Betriebssystem gehören, aber trotzdem vom Zugriff aller Anwendungen geschützt werden sollen; und auf Ebene 3 (CPL=3) befinden sich die einzelnen Anwendungen. Dabei gilt:

- Programme niedriger Privilegebenen können auf Datensegmente in höheren Privilegebenen nicht zugreifen.
- Der Zugriff eines Programms auf Datensegmente, die sich entweder auf der gleichen oder einer niedrigeren Privilegebene aufhalten, ist jedoch erlaubt.
- Auf Codesegmente einer höheren Privilegebene kann nur indirekt mit Hilfe eines für das Programm transparenten Steuermechanismus, eines sogenannten *Gates* zugegriffen werden. *Gates* geben feste Einsprungadressen vor, wodurch ein unkontrollierter Zugriff auf höher privilegierte Ebenen ausgeschlossen ist.
- Auf Codesegmente, die auf der gleichen Privilegebene liegen wie das Programm, das sie nutzen möchte, darf zugegriffen werden.
- Auf Codesegmente, die auf einer niedrigeren Privilegebene liegen als das Programm, das sie nutzen möchte, darf nicht zugegriffen werden, da der Vertrauensgrad dieses Codes geringer ist als der des aufrufenden Programms.

Es ist nicht notwendig, alle Privilegebenen zu benutzen. OS/2 zum Beispiel benutzt nur drei, nämlich die Ebenen 0, 2 und 3.
Auf der Ebene der Pages existieren außerdem zwei Privilegebenen: Ein Supervisorlevel, für Programme des Betriebssystems, Systemdaten wie Page-Tables usw; und ein Userlevel für Daten und Code der einzelnen Anwendungen. Tasks, die auf Ring 0, 1 oder 2 laufen, agieren grundsätzlich auf Supervisorlevel.

. Der virtuelle 8086-Modus

DOS-Anwendungen laufen im Real Mode. Um sie in einem Protected Mode-System zu starten, muß der Prozessor entweder in den Real Mode umschalten (wie der 80286) oder eine entsprechende Umgebung simulieren, wie der 80386 es tut. Der 80386 ermöglicht die Ausführung mehrer 8086-Anwendungen im Protected Mode. Eine 8086-Anwendung läuft daher innerhalb einer sogenannten virtuellen 8086-Task, die wie jede Protected Mode-Anwendung auch auf Ring 3 ausgeführt wird.

Eine virtuelle 8086-Task stellt eine sogenannte virtuelle 8086-Maschine bereit, welche virtuelle Register (mit Hilfe des TSS dieser Task) und einen virtuellen Adreßraum (das 1 Mbyte im linearen Adreßraum) zur Verfügung stellt. Auch die I/O-Systemkomponenten werden in der virtuellen 8086-Task simuliert. Alle Zugriffe auf diese Einheiten werden vom Betriebssystem gesteuert. Wenn eine 8086-Anwendung z.B. einen Hardware-Interrupt auslöst, so wird er durch das Betriebssystem abgefangen und durch geeignete Programme entsprechend verarbeitet. Auf diese Weise merkt das 8086-Programm nicht, daß es auf einer komplett simulierten 8086-Maschine läuft, während das Betriebssystem dafür sorgt, die virtuelle 8086-Task in die Protected Mode-Umgebung zu integrieren, um die sichere Betriebsumgebung nicht zu gefährden. OS/2 macht vom virtuellen 8086-Modus des 80386 Gebrauch und stellt damit eine vorzügliche Umgebung für DOS- und Windows 3.x-Anwendungen zur Verfügung. Unter OS/2 ist diese Unterstützung mit sogenannten VDMs (Virtual DOS Machines) realisert, die an jedes Programm individuell angepaßt werden können. Die Implementierung dieser VDMs ist recht komplex, so daß wir die Einzelheiten der DOS-Emulation unter OS/2 in einer späteren Ausgabe zusammen mit ausführlichen Tips zur Konfiguration vorstellen werden.

Die Folgemodelle des 80386

Zum Abschluß dieses Artikels möchten wir Ihnen noch einen kurzen Überblick zu den Prozessoren geben, die nach dem 80386 auf den Markt kamen.

Der Prozessor der nächsten Generation war der 80486. Wie sein Vorgänger verfügt er über einen 32-Bit breiten Daten- und Adreßbus und sämtliche Leistungsmerkmale des 80386, die wir ausführlich beschrieben haben. Neuerungen bestanden darin, daß auf dem gleichen Chip die Fließkommaeinheit (FPU) untergebracht war, also der Coprozessor. Beim 80386 mußte man den **Coprozessor** extra kaufen und in einen entsprechenden Steckplatz auf der Hauptplatine einbauen. Der Coprozessor ist notwendig, um Gleitkommazahlen zu verarbeiten. Die EU des Prozessors kann nämlich nur Ganzzahlen verarbeiten. Durch einen Coprozessor erhöht sich die Rechenleistung, weil man ohne diesen zweiten Rechner entsprechende Softwareemulationen zur Verfügung stellen muß, um Gleitkommaoperationen durchführen zu können. Unter OS/2 existiert dazu die eine Bibliothek namens NPXEMLTR.DLL.

Alle Prozessoren ab dem 80386 verfügen über eine **FPU**, die auf dem gleichen

Chip wie die CPU untergebracht ist[3]. Eine weitere Besonderheit war ein 8 KByte großer Instruction-Cache, mit dessen Hilfe die letzten Instruktionen zwischengespeichert werden können, auf die zuletzt zugegriffen wurde. Durch den internen Instruction-Cache wurde die Verarbeitungsgeschwindigkeit gegenüber dem 80386 erhöht. Zur Verwaltung dieses Caches dienen 6 neue, 80486-spezifische Befehle. Alle dem 80486 folgenden Modelle verfügen ebenfalls über einen solchen internen Cache, der allerdings größer als 8 KByte ist. Bereits die späteren 80486-Prozessoren verfügten über einen 16 KByte großen internen Cache.

Die letzte Veränderung gegenüber dem 80486 bestand in einer Erhöhung der Taktfrequenz, mit welcher der Prozessor betrieben werden konnte. Der schnellste 80386 war der 80386 DX-40 von AMD. Die schnellsten 80486 werden mit 100 oder sogar 133 MHz getaktet. Die hohe Takfrequenz wird durch einen sogenannten Clock-Doubler erzielt. Er verdoppelt, verdrei- oder vervierfacht den Systemtakt. Die CPU wird dann mit 66, 100 oder 133 MHz getaktet, der Systembus aber nur mit 33 MHz. Nach dem 80486 brachte Intel den Pentium-Prozessor auf den Markt, der neue Maßstäbe setzte. Heutzutage basieren alle Intel-kompatiblen Prozessoren auf der Pentium-Technologie.

Bei diesem Prozessor wurde nicht nur der Befehlssatz erweitert, sondern die ganze Architektur grundlegend überarbeitet. Dabei ist der Pentium vollkommen binärkompatibel zu allen seinen Vorgängern; das Speichermanagement hat sich auch hier nicht geändert, so daß der Pentium auf genau die gleiche Weise den Speicher adressiert wie der 80386. Der Pentium-Prozessor verfügt jedoch:

- über einen 64-Bit breiten Datenbus, um die Kommunikation mit dem Rest des System zu beschleunigen,
- eine völlig neu überarbeitete FPU, die über eine siebenstufige Pipeline verfügt; oft benutzte Befehle sind bereits hardwareseitig implementiert, um eine schnellere Ausführung zu erzielen. Daneben wurden neue und verbesserte Algorithmen aufgenommen, womit eine erhebliche Leistungssteigerung gegenüber der 486-FPU entsteht,
- zwei getrennte 8 KByte große interne Code- und Datencaches,
- eine Einheit zur Verzweigungsvorhersage (Branch Prediction), mit welcher die am meisten benutzten Befehle analysiert werden um zu bestimmen, welche Sprungbefehle innerhalb des Programmcodes am wahrscheinlichsten auftreten werden (die errechneten Wahrscheinlichkeiten liegen dabei bei über 88-90 %). Auf diese Weise kann immer für eine volle Befehlsqueue für die EU gesorgt werden. Für die Verzweigungsvorhersage stehen zwei Caches zur Verfügung, in denen der für die Ausführung vorgesehene Code zwischengespeichert wird,

---

3 Eine Ausnahme waren die 486SX-Prozessoren, die wie der 80386 ohne integrierte Fließkommaeinheit ausgeliefert wurden.

- die Implementierung von RISC-Strukturen. Der 80386 basiert auf der CISC-Architektur (Complex Instruction Set Computer), die über einen großen Befehlssatz verfügen, um eine einfache Programmierung zu gewährleisten. Im Gegensatz dazu steht die RISC-Architektur (Reduced Instruction Set Computer), die einen kleineren Befehlssatz aufweist, wobei die einzelnen Befehle flexibler zusammengesetzt werden können. Damit sinkt der Decodieraufwand, wodurch sich die Verarbeitungsleistung erhöht. Der Pentium-Prozessor verfügt zwar über einen noch höheren Befehlssatz als der 80486, jedoch sind in ihm einige typische RISC-Merkmale implementiert,

- eine superskalare Architektur, wobei zwei EUs parallel arbeiten. Beide EUs können pro Taktzyklus einen Befehl aufnehmen, womit zwei Instruktionen gleichzeitig verarbeitet werden können.

Eine weitere Besonderheit besteht in einer Option, welche die Größe der Pages betrifft. Der Pentium-Prozessor bietet optional auch eine Page-Größe von 4 MByte an, um ein häufiges Auslagern von Pages auf die Festplatte zu vermeiden. Um so große Pages zu verarbeiten, muß ein System allerdings von vornherein über genügend Hauptspeicher verfügen, daneben werden solche Pages rasch unhandlich und lassen durch ihre Größe keine so optimale Benutzung des Speichers zu wie kleinere, 4 Kbyte große Abschnitte. Der Petium wurde zunächst mit 60 und 66 MHz ausgeliefert. Die 100 MHz-Grenze ist jedoch schon lange überschritten worden.

Der Nachfolger der Pentium-Prozessoren seitens Intel war zunächst der Pentium Pro-Prozessor und schließlich der Pentium II. Die Prozessoren wurden kontinuierlich mit neuen Befehlssätzen erweitert (etwa dem MMX-Befehlssatz). Daneben wurden die internen Caches weiter vergrößert und die Taktfrequenz konnte ebenfalls erhöht werden. Auf der Pentium-Technologie basierende CPUs werden auch von AMD (der K5 und K6, ab nächstes Jahr der K7) und Cyrix (die 5x86 und 6x86-Prozessoren, bzw. der MII) angeboten. Sie sind mit den Intel-Prozessoren kompatibel und bieten die gleichen, zum Teil bessere Leistungen zu einem meist geringeren Preis als die Intel-Pendants. Besonders über diese neuen Konkurrenzprodukte im Vergleich zu den entsprechenden Intel-Modellen und ihre Leistung unter OS/2 werden wir in den nächsten beiden Ausgaben berichten, so daß wir uns eine knappe technische Vorstellung dieser Prozessoren für ihren Test in der Praxis vorbehalten.
Wichtig jedoch ist, daß diese Prozessoren trotz ihrer Verbesserungen immer noch nach genau den gleichen Prinzipien arbeiten wie der 80386. Damit ist auch die volle Kompatibilität mit OS/2 gewährleistet, selbst wenn man auf den Prozessoren Windows-Logos finden sollte, die eine Optimierung für Microsoft-Produkte signalisieren. ❑

Zip Drives in IDE und Parallel

Die *Zip Drives* von *Iomega* sind wohl jedem zumindest vom Hören bekannt. Seit der Markteinführung im Jahre 1995 sind diese Laufwerke mit den etwa diskettengroßen Medien zu einer Art Standard avanciert, und man findet sie fast überall. Sie bieten eine gute Alternative zur herkömmlichen Diskette und eignen sich als zuverlässiges und schnelles Backup-Medium ebenso gut wie für das Austauschen größerer Datenbeständen zwischen verschiedenen Systemen.
Zip-Laufwerke gibt es für den Parallelport, für IDE und SCSI. Wir haben uns zunächst den ersten beiden Laufwerkstypen gewidmet und ihre Tauglichkeit unter OS/2 getestet. Dazu betrachteten wird das externe *ZIP 100 parallel port* und das interne *Zip 100 ATAPI*-Laufwerk von *Iomega* unter *Warp 3* und *Warp 4*.

Lieferumfang

Hat man den Schutzkarton entfernt, findet man die beiden Laufwerke in eierkartonähnliche Pappdeckel eingepackt; was die Umwelt betrifft, so denkt Iomega also mit. Der Umschlagkarton stimmt einen OS/2-Anwender allerdings nicht sehr optimistisch: Gleich drei Windows-Logos findet man darauf abgebildet, von OS/2-Kompatibilität auf der Verpackung also keine Spur (Altpapier läßt sich allerdings gut entsorgen). Da man auf beide Laufwerke 1 Jahr Garantie hat, sollte man damit allerdings noch warten.
Man findet zu beiden Laufwerken ferner ein recht umfangreiches Handbuch, einige Beilagen, Tool- und Treibersoftware und Registrierkarten. Beim Parallel-Laufwerk ist eine Diskette mit Treibern für Windows und DOS sowie eine Zip-Disk mit Werkzeugen enthalten; beim IDE-Zip findet man sogar eine CD, auf der man Tools für alle Windows-Versionen findet. Sowohl die CD als auch die Diskette bzw. die Zip-Disk braucht man nicht, ebenso wenig wie die Installationshinweise in den Handbüchern (bis auf die Abschnitte, die sich mit der Hardwareinstallation beschäftigen). Außer Tips zur Installation der Laufwerke unter den Windowsderivaten findet man darin nichts, was für OS/2 relevant wäre. Es ist natürlich ärgerlich, daß sich der Hersteller nicht die wenig belastende Mühe macht, die nötigen Treiber und Installationshinweise für andere Plattformen ebenfalls direkt mit dem Laufwerk zu liefern. So ist man wieder auf sich allein gestellt. Immerhin gibt es von Iomega zwei OS/2-Treiber und einige Werkzeuge. Was Sie außerdem noch zum Betrieb der Laufwerke benötigen, haben wir in einer Tabelle am Ende dieses Artikels zusammengefaßt.

Das Laufwerk für den Parallelport

Neben der beschriebenen Software findet man im Lieferumfang des Parallelportlaufwerkes neben dem Laufwerk selbst ein Netzteil und ein paralleles Kabel. Mehr braucht man zum Betrieb nicht. Den Eingang für dieses Kabel findet man auf der Rückseite des Laufwerkes, gleich daneben einen parallelen Ausgang für den Anschluß des Druckers. Diesen Ausgang benötigt man, wenn man Drucker und Laufwerk am gleichen Port verwenden möchte, was übrigens mit beiden OS/2-Versionen und mit verschiedenen

System ohne Schwierigkeiten funktionierte.
Das Laufwerk selbst ist recht robust verarbeitet. Es besitzt sowohl an der Unterseite als auch an einer der Außenseiten Gummifüßchen, die einen rutschsicheren Halt auf dem Schreibtisch oder dem Computergehäuse ermöglichen. Die Buchse für das vom Netzteil kommende Kabel findet man an der rechten Laufwerkseite, wo es in einem kleinen Schlitz verborgen werden kann. Der Auswurf für das Medium befindet sich gut zugänglich rechts an der Vorderseite des Laufwerks, darüber sind zwei LEDs, eine für die Spannung (grün) und die andere zum Registrieren von Aktivitäten des Laufwerkes (orange).
Soweit ist das externe *Zip* eine feine Sache, weist allerdings auch zwei unangenehme Mängel auf: Einmal ist das Parallelkabel viel zu kurz, es mißt gerade mal an die 40 cm. Nicht genug, wenn man es auf dem Schreibtisch unterbringen möchte. Wir haben uns beholfen, indem wir es auf die Oberseite eines Towergehäuses stellten, allerdings kommt man dann weniger gut an das Laufwerk heran. Man ist also gut beraten, sich beim Bezug des Laufwerkes ein längeres LapLink-Kabel zu kaufen, um ein wenig freier in der Wahl des Aufstellungsortes zu sein. Der Betrieb mit einem solchen Nicht-Iomega-Kabel verursachte keine Probleme.
Ein weitaus schwerwiegenderer Nachteil ist das Fehlen eines Schalters für die Spannungsversorgung. Wohl oder übel hängt das Laufwerk ständig am Netz. Das ist nicht sehr angenehm, einmal weil es ununterbrochen Strom verbraucht; zum anderen weil jedes Ein- und Ausschalten mit einem Kriechgang unter den Schreibtisch verbunden ist. Das Jaz-Drive ist da etwas fortschrittlicher und hat einen Schalter an der Rückseite. Daß *Iomega* dem *Zip* nicht auch einen solchen Schalter gönnte, ist schade. Wir haben uns mit einer Steckdosenleiste beholfen, die man ein- und ausschalten kann. Eine Anschaffung, die sich im übrigen für jedes Rechnersystem lohnt.

Treiber

Mit dem *ZIP parallel port* gab es bezüglich des Treibers keinerlei Probleme, da der Hersteller ein entsprechendes Treiberpaket bietet und zwar gleich in zwei Versionen: Einen ADD mit einem entsprechendem Filter für die SCSI-Varianten (*Zip* und *Jaz*) und einen OAD (Open Architecture Driver) ebenfalls für den Betrieb der SCSI-Typen und auch des Parallelportlaufwerkes. Zu entscheiden, welchen dieser Treiber man verwendet, ist nicht ganz leicht, zumal die Dokumentation des Herstellers noch mehr Unklarheit schafft. Für das *Zip parallel port* brauchen Sie auf jeden Fall nur den OAD. Die Herstellertreiber finden Sie auf unserer Homepage.
Dieser Treiber, der in der Version 2.34 vorliegt, ist gut und gern drei Jahre alt, neuere Versionen gibt es nicht. Allerdings verspricht er eine problemlose Installation und ein angenehmes Arbeiten mit dem Laufwerk. Welche Laufwerke dieser und der ADD-Treiber unterstützt, finden Sie ebenfalls in einer Tabelle am Ende dieses Berichtes. Bis auf das *Zip parallel port* werden die Herstellertreiber allerdings nicht oder nur bedingt benö-

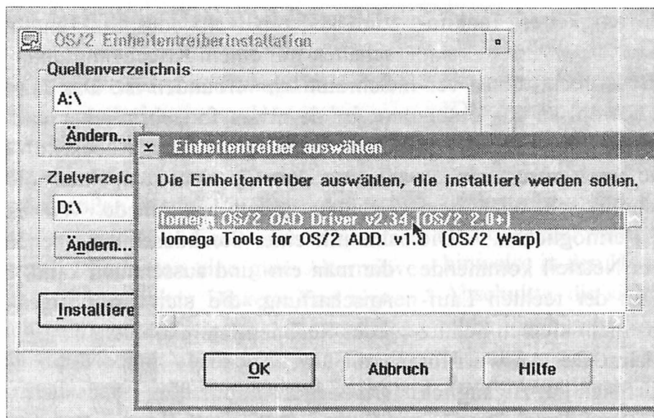


Abb.1: Den Iomega OAD Treiber installieren

tigt. Alle anderen Laufwerke von Iomega, wie das *Zip ATAPI* lassen sich auch mit dem betreiben, was OS/2 bietet (wenngleich das der Hersteller nicht weiß).

Installation

Die Einrichtung des Laufwerkes im System ist, wie man sich denken kann, problemlos: Man schließt es mit dem beiliegenden Kabel einfach an einen freien Parallelport an. Läßt man das Laufwerk über LPT1 laufen, verbindet man das Druckerkabel über die beschriebene Buchse mit dem Laufwerk.

Nach erfolgreichem Anschluß geht es an die Installation des Treibers, die ebenso einfach ist. Gehen Sie dazu wie folgt vor:

❶Besorgen Sie sich den Treiber wie unter Treiberunterstützung angegeben. Auf unserer Homepage finden Sie ihn auf der *Treiberseite* der *OS/2 Only! Downloadpage* als Datei mit dem Namen *os2v234.exe*. Es handelt sich dabei um eine selbstentpackende DOS-Datei.

❷Entpacken Sie diese Datei in einem temporären Verzeichnis (über die Kommandozeile os2v234 eingeben; mit der WPS einfach auf das os2v234-Objekt doppelklicken).

Nach dem Entpacken finden Sie eine Readme-Datei, eine Zipdatei namens O2345.ZIP, sowie ein PKUNZIP für DOS und eine EXTRACT.BAT zum Entpacken. Entpacken Sie die ZIP-Datei durch Doppelklick auf die EXTRACT.BAT oder mit einem OS/2-Unzip (finden Sie ebenfalls auf unserer Homepage). Dadurch werden zwei Verzeichnisse namens DISK1 und DISK2 erstellt. DISK1 enthält die Iomega-Tools für Win-OS/2; DISK2 die angesprochenen Treiber. Verschieben Sie den Inhalt beider Verzeichnisse auf zwei entsprechend beschriftete Disketten. Sie benötigen im folgenden nur die Treiberdiskette.

❸Überprüfen Sie, ob das Laufwerk mit Spannung versorgt wird und korrekt am Parallelport angeschlossen ist. Legen Sie die Diskette mit den Iomega-Treibern in das Laufwerk A: und starten Sie das Dienstprogramm *Einheitentreiber installieren*. Sie finden es unter Warp 3 im Ordner *Systemkonfi-*

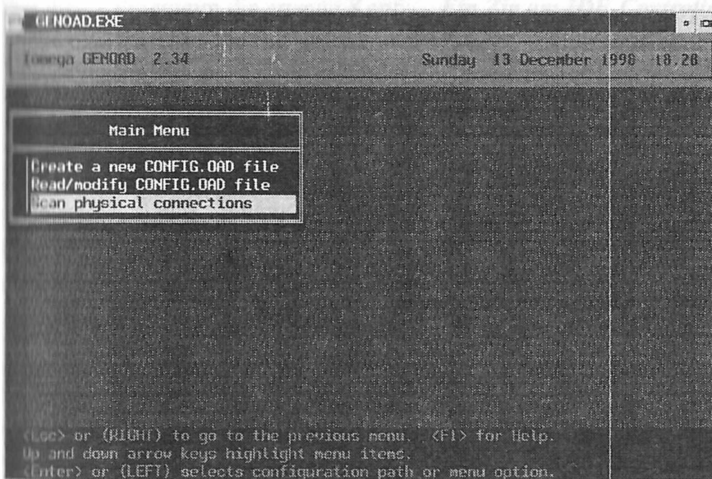


Abb. 2: GENOAD nach dem Laufwerk am Parallelport suchen lassen

guration, der sich im *System*-Ordner auf der Arbeitsoberfläche befindet, bzw. unter Warp 4 im Ordner *Installieren/Entfernen,* welcher im Ordner *Systemkonfiguration* liegt. Wenn Sie die WPS nicht verwenden, starten Sie das Programm über die Kommandozeile. Geben Sie dazu den Befehl DDINSTAL ein.

• Drücken Sie nach dem Start des Programms *Installieren*. Das Programm zeigt Ihnen zur Auswahl den *Iomega OS/2 OAD Driver v2.34* und Tools für den ADD-Treiber (siehe Abb. 1). Markieren Sie den ersten Eintrag und klikken Sie auf *OK*. Das Programm erzeugt dann auf dem Installationslaufwerk das Verzeichnis OAD, kopiert alle nötigen Dateien und trägt den OAD-Treiber namens OS2.SYS in die CONFIG.SYS ein. Wenn das Programm beendet ist, bestätigen Sie alle Hinweise, führen Sie einen Systemabschluß durch und starten Sie den Rechner neu. Wird der Treiber beim Systemstart das erste Mal gealden, meldet er sich mit mehrmaligen Piepen, da er noch nicht konfiguriert ist. Das ist allerdings völlig normal.

Tip: *Das Programm Einheitentreiber installieren fügt das OAD-Verzeichnis nicht in die PATH-Anweisung der CONFIG.SYS ein (die Anweisung fehlt in der Profilsteuerdatei). Bevor Sie also den Rechner neu starten, editieren Sie am besten die CONFIG.SYS, um den Eintrag nachzuholen. Sie müssen später Programme aus diesem Verzeichnis aufrufen, so daß es praktisch ist, nicht immer das Verzeichnis wechseln zu müssen.*

• Nach dem Neustart müssen Sie den Treiber konfigurieren. Öffnen Sie dazu ein OS/2-Fenster oder einen OS/2-Gesamtbildschirm. Wechseln Sie in das Verzeichnis \OAD auf dem Installationslaufwerk, wenn das Verzeichnis

nicht im PATH eingetragen ist, und starten Sie GENOAD.EXE. Nachdem das Programm gestartet ist, drücken Sie *ESC*, um den Begrüßungsbildschirm zu entfernen und wählen aus dem Menü mit den Pfeiltasten *Scan Physical Connections*[4] (Abb.2) . Das Programm sucht dann nach dem angeschlossenen Laufwerk (was etwa 10 bis 15 Sekunden dauert) und zeigt es mit dem zugehörigen Treiber in einer Liste an (Abb 3).

! ***Ausgeschaltete oder nicht angeschlossene Laufwerke erkennt GENOAD nicht. Findet GENOAD das Laufwerk nicht, überprüfen Sie zunächst die Spannungsversorgung, die Verbindung zum Parallelport und schließlich den Parallelport selbst.***

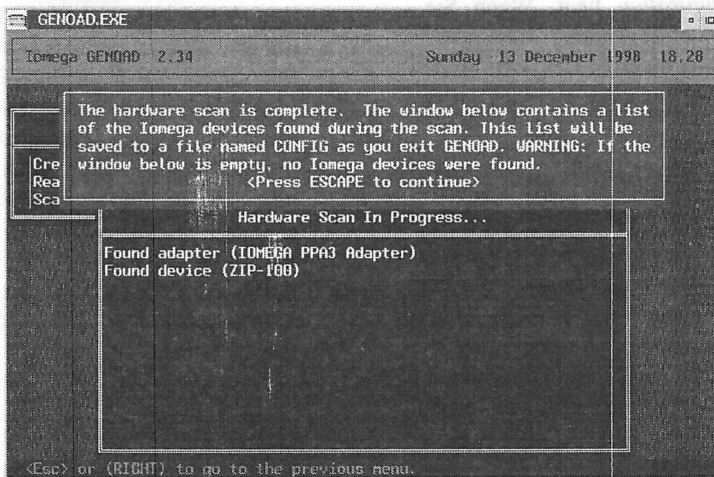


Abb. 3: GENOAD nach dem Hardware Scan

Nachdem der Scan abgeschlossen ist, drücken Sie wieder *ESC* und bestätigen bei der Abfrage *Save OAD Changes* mit der Auswahl des Feldes *Yes*. Bestätigen Sie die Abfrage *Overwrite CONFIG* ebenfalls mit *Yes* und beenden Sie das Programm durch Drücken von *ESC* und der erneuten Auswahl von *Yes*.

❷Führen Sie einen Systemabschluß durch, und starten Sie den Rechner neu. Beim nächsten Neustart zeigt der OAD-Treiber beim Laden *Scanning OAD Configuration File...* an und meldet schließlich die Laufwerksbezeichnung, unter dem das ZIP geführt wird, z.B. *Supporting Logical Drive K:*

Damit ist die Installation abgeschlossen. Sowohl von der Kommandozeile aus als auch auf der Arbeitsoberfläche steht Ihnen nun das ZIP am Parallelport als neues Laufwerk zur Verfügung.

Schließen Sie irgendwann weitere ZIP-Laufwerke an Ihr System an, starten Sie GENOAD und nehmen

4 Menüpunkte werden in GENOAD mit Enter ausgewählt. Zum Fortfahren muß ESC gedrückt werden. Die Benutzung der ESC-Taste ist daher etwas ungewöhnlich.

über dieses Programm die erneute Konfiguration wie oben angegeben vor. Die beschriebene Vorgehensweise bereitete keine Probleme mit beiden Betriebssystemversionen und funktioniert tadellos sogar unter OS/2 2.1, so daß Sie evtl. auch einen älteren Rechner, auf dem diese Version noch läuft, mit einem *Zip-Laufwerk* ausstatten können (etwa ein altes Notebook, dem es oft an Speicherplatz fehlt).

Tip: *Wenn Sie mit der WPS arbeiten, geben Sie dem neuen Laufwerksbuchstaben einen aussagekräftigen Namen, etwa ZIP Drive, um es von den anderen klar zu unterscheiden.*

Betrieb

Im Betrieb präsentiert sich das ZIP-Drive wie ein großes Diskettenlaufwerk. Mehr zur Umgehensweise braucht man nicht zu erklären. Die WPS bietet für das neue Laufwerksobjekt unter anderem die Option *Datenträger auswerfen*, die im übrigen auch den gewünschten Effekt erzielt. Wer nicht mit der WPS arbeitet, kann das gleiche mit dem Iomega Utility LOCK.EXE (im Verzeichnis \OAD) und dem Parameter /e über die Kommandozeile erreichen.

Anzumerken ist aber, daß das externe ZIP-Laufwerk den Datenträger im wahrsten Sinne des Wortes auswirft. Steht das Laufwerk also an einem Platz, von dem aus das Medium zu Boden fallen kann, sollte man vorsichtshalber die Hand vor den Ausgabeschlitz halten.

Ein Zip am IDE-Controller

IDE ist die Lösung in Sachen Massenspeichersysteme, die sich auf dem Endanwendermarkt durchgesetzt hat. Das heißt nicht, daß es das bessere System ist; aber es ist kostengünstiger und für den Einstieg nicht verkehrt. Ferner gibt es eine Fülle unterschiedlichster Geräte für IDE, und so ist auch der OS/2-Anwender versucht, entsprechend »exotische« Laufwerke an einen IDE-Controller zu hängen. Mit dem *Zip 100 ATAPI* Laufwerk geht es, wenn auch mit einigen Hindernissen.

Das Laufwerk wird mit allem geliefert, was für die Hardwareinstallation nötig ist. Dabei sind ein IDE-Kabel, ein Y-Kabel für das Netzteil, ein 5¼"-Einbaurahmen, denn das Zip ist nur 3½" groß, und ein Tütchen mit vier Schrauben zum Einbau. Die Dokumentation ist sehr gut. Sie gibt die einzelnen Schritte beim Einbau des Laufwerkes in das System wieder. Wie das Zip an den IDE-Port angeschlossen werden kann, ist dem Handbuch ohne Mühen zu entnehmen.

Das Laufwerk macht einen robusten Eindruck - bis auf die Anschlußleiste am hinten Teil, welche das IDE-Kabel aufnimmt. Man hätte sich diese Steckerleiste etwas stabiler am Laufwerk angebracht gewünscht. Jedoch ist dieser Nachteil in der Verarbeitung zu verkraften, da man ja nicht jeden Tag das Kabel herauszieht, um es wieder hineinzustecken, so daß diese Stelle auch nicht unnötig beansprucht wird. Sollte sie aber besser auch nicht. Beim Einstecken des Kabels ist demnach ein wenig Vorsicht angesagt.

Das IDE-Kabel, das Iomega mitliefert, ist übrigens lang genug, um es durch selbst

große Gehäuse an anderen Laufwerken vorbeizuführen, sollte man wenig Platz haben. Die Jumper, mit denen man das Laufwerk am IDE-Port konfiguriert, sind leicht zugänglich am hinteren Teil des Laufwerkes angebracht. Der Installation der Hardware steht also nichts im Wege.
Die Auswurftaste am Laufwerk, sofern man sie denn braucht, denn die Auswurffunktion läßt sich auch über die Software steuern (und wenn man mit HPFS-formatierten Medien arbeitet, geht das auch gar nicht anders, da OS/2 in einem solchen Falle das Laufwerk sperrt), findet man in der LED, welche die Aktivitäten des Zips anzeigt, an der Vorderseite des Laufwerkes. Es ist ein wenig ungewohnt, auf eine LED zu drükken, so daß man vielleicht zunächst vergeblich nach einer Eject-Taste sucht.
Das interne ATAPI-Zip wirft die Medien weniger kraftvoll aus als das externe Pendant am Parallelport. Manchmal ist die Wucht aber doch größer, und das Medium fällt vom Gehäuse auf den Boden (bei unserem Towergehäuse ein tiefer Fall). Also sollte man beim Auswerfen eines Mediums auch hier vorsichtshalber die Hand vor das Laufwerk halten, um ein Fallen der Disk auf den Boden verhindern zu können. Bis auf diese kleinen aber nicht so tragenden Mängel kann man gegen das interne *Zip-Laufwerk* allerdings nichts sagen.

Treiber

Hatte man mit dem Laufwerk für den Parallelport keine Probleme, einen Treiber zu bekommen, kann einem das IDE-Zip wahre Kopfschmerzen bereiten. Es läßt sich zwar problemlos in nahezu jeder Hardwarekonfiguration im System unterbringen, aber daß heißt noch lange nicht, daß OS/2 es auch erkennt. Einen Treiber findet man beim Hersteller nicht. Auch in den Laufwerksspezifikationen des ATAPI-Zips von Iomega taucht OS/2 unter *Operating System Compatibility* nicht auf. Ein Anruf bei der Iomega-Hotline war noch enttäuschender, weil man dort mit OS/2 nicht sehr viel anzufangen wußte. Dort wurde gesagt, es gäbe »keine Chance«, das Laufwerk unter OS/2 zu betreiben. Nur um es vorwegzunehmen: Es läßt sich betreiben!
Mit dem Fixpak 35 für Warp 3 und dem Fixpak 6 für Warp 4 hat sich an der Unterstützung großer EIDE-Festplatten, wechselbarer Medien und am HPFS-Dateisystem einiges getan: Zum einen werden EIDE-Laufwerke bis zu einer Größe von 8,4 GByte unterstützt; zum anderen können ATAPI-Wechsellaufwerke wie das *Iomega Zip* betrieben werden. Die Unterstützung für IDE- und EIDE-Geräte bietet der IBM1S506.ADD-Treiber; für ATAPI-Laufwerke der IBMATAPI.FLT. Besonders erfreulich ist, daß die Medien mit HPFS formatiert werden und gefahrlos gegen andere ausgewechselt werden können. Dazu bieten die WPS-Objekte für Laufwerke mit wechselbaren Medien wie dem *Zip* eine neue Auswurf-Funtkion, die ein gefahrloses Entnehmen des Mediums ermöglicht; für die Kommandozeile gibt es das kleine Programm EJECT, das genau dasselbe bewirkt.
Also heißt es fixen, wenn man das Iomega ATAPI *Zip* unter OS/2 betreiben möchte. Die beiden angegebenen Fixpaks standen uns zur Verfügung. Wir spielten

sie daher auf und fügten ein weiteres Fix hinzu, das Sie sich von unserer Homepage ziehen können (Datei: ideatapi.zip). Sollten Sie Ihr Warp mit einem aktuellen Fixpak als 35 für Warp 3 und 6 für Warp 4 betreiben, brauchen Sie laut IBM diese Datei nicht. Diejenigen, welche allerdings mit Warp 3 mit Fixpaks unter 35 oder Warp 4 mit Fixpaks unter 6 arbeiten, sollten zunächst diese Fixpaks aufspielen und anschließend das ATAPI-Zip installieren. Von einer Installation des IDE-fixes auf eine andere Warp-Version unter Fixpak 35 (bzw. 6) ist abzuraten. Wir haben es ausprobiert und damit schlechte Erfahrungen gemacht (eingeschränkte Funktionalität, HPFS-formatierte Medien konnten nicht gewechselt werden, bis hin zu einem Trap, hervorgerufen durch eine Exception im IBM1S506.ADD). Also: Erst fixen, dann das Laufwerk in Betrieb nehmen. Wo es die Fixpaks gibt, erfahren Sie im Kapitel Neuigkeiten.

Installation

Die Hardwareinstallation ist sehr gut im Handbuch beschrieben, so daß wir hier nicht genau darauf eingehen müssen. Nur zu den Konfigurationen: Das IDE-Zip kann als Slave oder Master an den IDE-Port angeschlossen werden. Wir haben es als Master am sekundären IDE-Controller betrieben; aber auch als Slave zusammen mit dem CD-ROM am sekundären Port oder als Slave am primären in Zusammenarbeit mit der Festplatte gibt es im Betrieb keine Schwierigkeiten. Sie können das Laufwerk also mit allen im Handbuch beschriebenen Hardwarekonfigurationen einsetzen. Standardmäßig ist das *Zip* als Slave gejumpert, so daß man hier evtl. Änderungen vornehmen muß. Das *Zip* läßt sich sowohl in einen 5¼" als auch in einen 3½"-Einbauschacht einsetzen. Beim letzteren muß man nur den Einbaurahmen entfernen.

Was den Treiber anbelangt, so spielen Sie zunächst ein Fixpak (mindestens 35) für Warp 3 bzw. (mindestens 6) für Warp 4 auf. *Nach* erfolgreicher Installation des Fixpaks 35 bzw. 6 installieren Sie bitte außerdem noch das Fix FIXED IDE HARD DRIVE AND REMOVABLE MEDIA SUPPORT auf. Gehen Sie dazu wie folgt vor:

❶ Besorgen Sie sich die Datei von unserer Homepage (Datei: ideatapi.zip). Entpacken Sie sie direkt auf eine Leerdiskette.

❷ Nehmen Sie die Installation über das Dienstprogramm Einheitentreiber installieren vor, wie es bereits beschrieben worden ist.

❸ Führen Sie anschließend einen Systemabschluß durch und starten Sie das System neu.

Wie bereits gesagt, ist bei höheren Fixpaklevels dieses Fix nicht erforderlich.

! ***Sollten Sie das Fix nicht benötigen, achten Sie bitte darauf, daß der Treiber IBMATAPI.FLT in der CONFIG.SYS eingetragen ist, denn OS/2 tut das nicht automatisch. Falls kein Eintrag vorhanden ist, fügen Sie ihn mit der Anweisung BASEDEV=IBMATAPI.FLT hinzu.***

Es sind keine weiteren Konfigurationsschritte zur Einrichtung des OS/2-Standardtreibers nötig. IBM1S506.ADD bestimmt die Hardwarekonfiguration automatisch. Wir stellten fest, daß nach der Installation des *Zip ATAPI* Laufwerkes das System etwas länger braucht, um den Treiber zu initialisieren, wodurch die Bootzeit ebenfalls steigt. Sollte es also scheinbar nach dem Laden des IBM1S506.ADD nicht weitergehen, so haben Sie einfach etwas Geduld.
Der Treiber arbeitet zuverlässig ohne Ausfälle und Probleme. Die Unterstützung der Laufwerke seitens IBM ist also nicht zu beanstanden und verleiht dem OS/2-Anwender eine noch höhere Flexibilität in der Wahl seiner EIDE- bzw. ATAPI-Einheiten. Wir werden die neue Version des Standardtreibers auch noch mit großen EIDE-Festplatten testen, um eventuelle Probleme ausfindig zu machen.

Leistung
Wir testeten die beiden Zips mit zwei verschiedenen Rechnern: Einem 6x86 P166 PCI-Rechner mit 32 MB RAM und einer 1,2 GB-EIDE-Festplatte von Connor und einer 486 DX/2-66 VLB-Maschine mit 16 MB RAM und einer 850 MB IBM-Festplatte. Beim Test der Geräte formatierten wir die Zip-Medien zunächst nur mit FAT.
Mit FAT wird das Zip zur Schnecke. Wir ermittelten durch Dateioperationen (Verschieben, Kopieren) eine Datentransferrate von etwa 85 KB/sek für das *ZIP parallel port*. Das ATAPI-Laufwerk brachte es gerade mal auf durchschnittlich 98 KB/sek. Das ist nicht gerade viel. Eine geminderte Geschwindigkeit wäre ja noch vertretbar, aber das Antwortverhalten der Systeme veränderte sich beim Betrieb der beiden Laufwerkes so sehr, daß man nicht mehr vernünftig damit arbeiten konnte. Auf dem älteren Rechner konnte man nebenbei einen Kaffee trinken, und selbst mit dem schnellen 6x86 ließen sich Fenster nur sehr schleppend verschieben und Programme brauchten viel zu lange um zu starten. Alleine um ein OS/2-Fenster zu öffnen, benötigte das System etwa 20 Sekunden, was kaum mehr vertretbar ist. Für ein umfangreiches Arbeiten mit den Laufwerken ist FAT als Dateisystem für die Zip-Medien also nicht geeignet.
Formatiert mit HPFS ist es jedoch eine wahre Freude, mit den Zip-Laufwerken zu arbeiten, weil man wirklich den Anschein hat, mit Festplatten umzugehen (selbst am Parallelport), nicht mit übergroßen Diskettenlaufwerken. Daneben merkt man von Datentransfers nicht viel, man kann ganz normal mit dem System weiterarbeiten, abzüglich der Einschränkungen, die IDE mit sich bringt. So brauchte das *ATAPI-Zip* im 6x86 1 Minute und 20 Sekunden, um 137 Dateien mit einer Gesamtgröße von 4,7 MB auf ein mit FAT formatiertes Zip-Medium zu kopieren; mit einer Zip-Diskette, welche mit HPFS formatiert wurde, dauerte das gerade mal 13 Sekunden! Den gleichen Effekt erzielten wir mit dem *Zip* am Parallelport: Der 486 übertrug 8,5 MB auf das externe Laufwerk mit HPFS-Medium in nur 83 Sekunden, und das System reagierte wie gewohnt auf Benutzereingaben. Nach diesen überraschenden Erfahrungen testeten wir bevorzugt

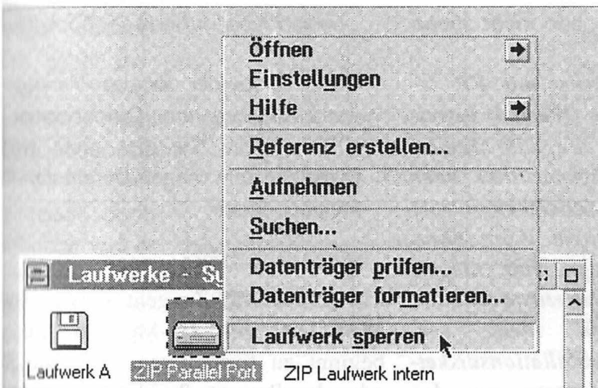


Abb. 4: Das Parallele Zip Laufwerk über die WPS sperren

den Umgang mit HPFS-formatierten Medien.

Vorbereiten der Datenträger
Wenn man die Zip-Medien kauft, sind sie mit FAT formatiert. Also muß man sie neu formatierten. Eine Partitionierung ist nicht erforderlich, da die Medien bereits partinioniert sind: Man findet auf ihnen eine primäre Partition. Das sollte auch so bleiben. Wir hatten Probleme mit logischen Partitionen auf einer Zip-Diskette. Allerdings braucht man keine Angst zu haben, daß sich durch die Installation der *Zips* oder durch das Erstellen einer primären Partintion auf einem Zip-Medium die Laufwerkbuchstaben ändern. Die wechselbaren Laufwerke werden von OS/2 automatisch an das Ende der Laufwerkliste gestellt. Sollte eine Partinionierung aber doch einmal erforderlich sein, funktioniert dies allerdings nicht mit dem Zip am Parallelport, da dieses Laufwerk nicht direkt von einem OS/2 ADD-Treiber unterstützt wird.
Um ein Zip-Medium, das in einem Laufwerk am parallelen Port eingelegt ist, für HPFS zu formatieren, gehen Sie wie folgt vor:

❶ Sperren Sie das Laufwerk. Verwenden Sie die WPS, so öffnen Sie das Kontextmenü des Laufwerkobjekts und wählen Sie Datenträger sperren (Abb 4). Von der Kommandozeile aus benutzen Sie das Iomega-Tool LOCK. Die Syntax hierfür lautet:

LOCK <Laufwerkbuchstabe:> /L

❷ Formatieren Sie das Laufwerk. Über die WPS öffnen Sie das Kontextmenü des Laufwerkobjektes und wählen *Datenträger formatieren* (Abb. 5). Im erscheinenden Dialog wählen Sie HPFS und starten den Formatiervorgang durch Drücken auf *OK*. Ist das Formatieren beendet, schließen Sie das Statusfenster durch Drücken auf *OK*.

! ***Wir hatten Probleme, ZIP-Medien im Parallelportlaufwerk mit Warp 3 und 4 für HPFS zu formatieren, wenn das Fixpak 35 (bei Warp 3) bzw. das Fixpak 6 (bei Warp 4) aufgespielt war. Der Formatiervorgang wurde entweder abgebrochen oder mit den Fehlermeldungen SYS1279 bzw. SYS0527 beendet. Mit einem ungefixten Warp 3 oder 4, bzw. mit OS/2 2.1 konnte eine HPFS-Formatierung jedoch durchgeführt werden. Formatiervorgänge mit den Fixpaklevels 38 (Warp 3) und 8***

(Warp 4) konnten wir nicht mehr testen.

Sollten Sie ein Warp mit FP 35 (Warp 3) oder FP 6 (Warp 4) betreiben, bereiten Sie die ZIP-Medien über eine Wartungspartition oder mit Hilfe von Notfalldisketten vor, wenn diese mit einem Warp ohne Fixes erstellt worden sind. Liegen solche Disketten oder entsprechende Wartungspartitionen nicht vor, benutzen Sie die Installationsdisketten des Betriebssystems, um das System zu starten und die Medien mit dem auf den Disketten bzw. den CDs befindlichen FORMAT vorzubereiten. Die Verwendung eines FORMAT einer »ungefixten« Warpversion auf einer solchen mit den angegebenen Fixpaklevels war im übrigen nicht möglich.

❸ Arbeiten Sie lieber mit der Kommandozeile, verwenden Sie zum Formatieren den Befehl Format:



Abb. 5: Zip Medien über die WPS formatieren

format <ziplaufwerk:> /FS:hpfs

OS/2 verwendet ab der Version 3.0 standardmäßig ein QuickFormat für HPFS. Wenn Sie dies nicht wollen, fügen Sie dem obigen Befehl den Parameter /L hinzu.

Mit dem *ATAPI-Zip* geht alles einfacher. Man legt ein Zip-Medium ein und beginnt zu formatieren, über die WPS oder den Prompt. Probleme irgend einer Art traten hier nicht auf. Man kann über das Kontextmenü des Laufwerkobjektes den Datenträger nach dem Formatiervorgang auswerfen, um einen neuen zu formatieren (siehe Abb. 6).

Handhabung von HPFS-ZIP-Medien

Der Gebrauch von mit HPFS-formatierten ZIP-Medien ist unkomplizierter als man denken mag. Das IDE-Laufwerkobjekt der WPS bietet zunächst keine Möglichkeit mehr zum manuellen Sperren (siehe Abb. 6). Das System sperrt die Laufwerke automatisch, nachdem ein Medium eingelegt worden ist. Damit ist die Auswurftaste am Laufwerk auch deaktiviert, so daß die Ausgabe des Mediums softwareseitig erfolgen muß. Der Menüpunkt Daten-

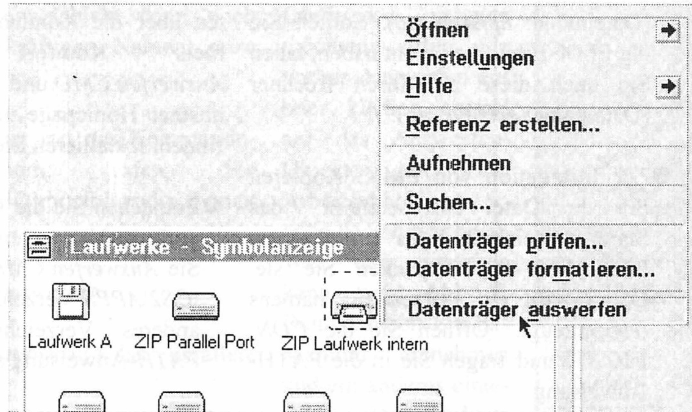


Abb.6: Datenträger des Zip ATAPI über die WPS auswerfen

träger auswerfen bleibt permanent funktionsfähig. Man kann daher ein HPFS-Medium einlegen, damit arbeiten und es auswerfen lassen; ein neues einlegen, die Arbeit fortsetzen und so fort. Es treten dabei auch keine Fehlermeldungen auf, etwa daß die Aktivitäten auf dem Laufwerk nicht beendet werden konnten, da OS/2 das Dateisystem auf das Entfernen des Mediums im Laufwerk vorbereitet. Auf der Kommandozeile kann man einfach mit EJECT arbeiten. Damit kommt man sehr komfortabel in den Genuß von HPFS auf dem internen IDE-Zip. Diese Neuerung des IBM-Fixpaks erfreute uns außerordentlich.

Allerdings - die Mühen, die man früher mit HPFS und den Wechsellaufwerken hatte, schwinden nicht ganz, was aber nicht an OS/2 liegt: Betreibt man nämlich ein Zip-Laufwerk am Parallelport, sieht die Sache ein wenig anders aus, da hier ein anderer Treiber die Steuerung des Laufwerkes übernimmt.

Die Laufwerke sind bekanntlich gesperrt, wenn man sie mit HPFS betreibt. Ein Auswurf über die Taste am Laufwerk ist daher nicht möglich. Sollte man nun mit LOCK das Laufwerk entriegeln und anschließend den Datenträger ausgeben, so wird man gezwungen sein, ein Checkdisk aufzurufen, wenn man die Zip-Disk wieder einführt. Ein erzwungenes softwareseitiges Auswerfen des Mediums führt zum gleichen Effekt. Also müssen einige Vorbereitungen getroffen werden. Dazu gibt es ein sehr nützliches Werkzeug namens *hpfsrem*. Das kleine Programm wird von der Kommandozeile aufgerufen und erlaubt die Vorbereitung eines mit HPFS formatierten Datenträgers zum Austausch gegen einen anderen[5]. Dieses Programm benötigt die EMX-Runtime-Bibliothek ab Version 0.9a. Auch viele andere Programme aus der Share- und Freewareszene benötigen diese Bibliothek, also schadet ein Download nichts. Um das Tool zu installieren, verfahren Sie wie folgt:

❶ Besorgen Sie sich die Software, am einfachsten von unserer Homepage

5 hpfsrem arbeitet nicht unter OS/2 2.1. Das Programm ist Shareware. Nach einem Testzeitraum muß es registriert werden.

(Dateiname: *hpfsrem.zip*). Sollten Sie die EMX-Bibliothek nicht haben, laden Sie auch diese auf Ihren Rechner (Dateiname: *emx09c.zip*).

❷Zur Installation von EMX: Kopieren Sie die Datei *emx09c.zip* in das Stammverzeichnis Ihres Installationslaufwerkes und entpacken Sie sie. Damit wird ein Verzeichnis namens */emx* erzeugt. Öffnen Sie die *CONFIG.SYS* und tragen Sie in die PATH-Anweisung *<Installationslaufwerk>:\emx\bin* ein; dem *LIBPATH* fügen Sie *<Installationslaufwerk>:\emx\dll* hinzu. Führen Sie einen Systemabschluß durch und starten Sie den Rechner neu.

❸Zur Installation von *hpfsrem*: Entpacken Sie die Datei *hpfsrem.zip* in einem temporären Verzeichnis. Darin finden Sie die *EXE*-Datei *hpfsrem.exe*, die Sie am besten in das *\OS2\APPS*-Verzeichnis verschieben oder in ein anderes Verzeichnis, das in der *PATH*-Anweisung angegeben ist.

Nun können Sie auch mit dem Parallellaufwerk HPFS-Medien verwenden. Zuvor richten Sie sich das neue Werkzeug aber noch etwas komfortabler ein. Wir haben dazu eine kleine Batchdatei geschrieben, die wir uns auf das *Launchpad* und das *WarpCenter* legten, um das Wechseln der Medien zu vereinfachen. Auch beim Arbeiten über die Kommandozeile bietet sie mehr Komfort. Sie heißt *Auswerfen.CMD* und ist ebenfalls auf unserer Homepage unter *auswurf.zip* zu finden. Installieren Sie sie wie folgt:

❶Entpacken Sie die Datei in einem temporären Verzeichnis und verschieben Sie *Auswerfen.CMD* am besten in das *\OS2\APPS*-Verzeichnis oder in ein anderes Verzeichnis, das in der *PATH*-Anweisung angegeben ist.

❷Arbeiten Sie mit der WPS erstellen Sie ein Programmobjekt. Öffnen Sie die Einstellungen zu diesem Objekt (Abb. 7). Tragen Sie auf der Seite Programm unter Pfad und Dateiname *Auswerfen.CMD* ein; im Eingabefeld Parameter den Buchstaben des Zip-Laufwerkes am Parallelport ein (mit Doppelpunkt, z.B. K:). Auf der Seite *Sitzung* versehen Sie den Eintrag *Beim Start Symbolgröße* mit einem Häkchen.

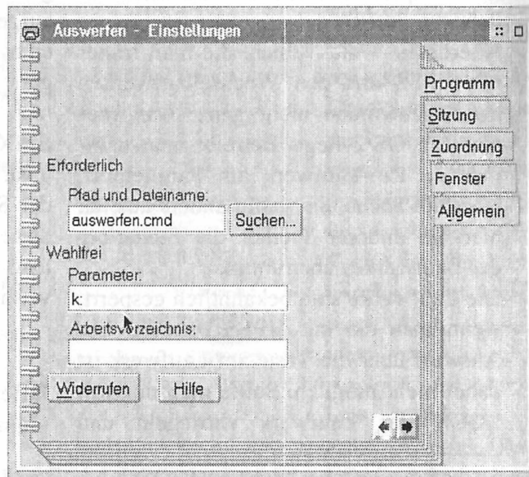


Abb. 7: Objekt für Auswerfen.CMD konfigurieren

Geben Sie dem Objekt dann noch einen aussagekräftigen Namen, etwa *Auswerfen*.

Wenn Sie lieber mit der Kommandozeile arbeiten, rufen Sie Auswerfen.CMD mit folgendie Syntax auf:

auswerfen <Laufwerk:>

Dabei ist *<Laufwerk:>* das parallele Zip.

! ***Auswerfen.CMD ruft hpfsrem und das Iomega-Tool LOCK auf. Achten Sie daher bitte darauf, daß diese beiden Programme in einem Verzeichnis stehen, das in der PATH-Anweisung angegeben ist. Normalerweise sorgt hpfsrem dafür, daß ein Laufwerk entriegelt wird, wenn es gesperrt ist. Das ist aber nicht immer der Fall. Deswegen ruft Auswerfen LOCK zweimal auf, um sicherzugehen, daß das Laufwerk entriegelt ist, bevor das Medium ausgegeben wird.***

! ***Rufen Sie Auswerfen.CMD nicht auf, wenn das Laufwerk abgeschaltet ist oder sich kein Medium im Laufwerk befindet.***

! ***Warten Sie mit dem Aufruf von Auswerfen.CMD, bis keine Aktivitäten auf dem Laufwerk mehr feststellbar sind.***

Und fertig! Immer wenn Sie eine Zip-Diskette aus dem externen Parallelport-Laufwerk nehmen möchten, rufen Sie vorher Auswerfen auf oder klicken einmal auf das Auswerfen-Objekt auf der Arbeitsoberfläche oder dem Launchpad. Wie das aussehen kann, zeigt Abb. 8. Dann können Sie das Medium entnehmen und ein anderes einlegen. Datenverluste treten nicht auf, womit ein sicheres Arbeiten mit HPFS auch mit dem Parallellaufwerk gewährleistet ist.

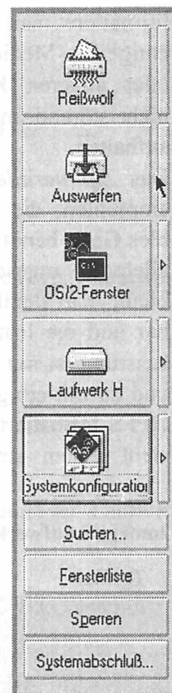


Abb. 8: HPFS-formatierte Zip Medien komfortabel über das Warp Launch-Pad auswerfen.

Das Letzte

Die beiden Laufwerke haben überzeugt. Sieht man mal von den ein oder anderen kleinen Mängeln in der Verarbeitung ab, so stellen sie doch wertvolle Helfer dar. Die Möglichkeit, schnell große Datenmengen auf den Zip-Medien festzuhalten, die eine höhere Datensicherheit haben als Disketten, macht sie zu empfehlenswerten Zusatzausstattungen.

Für den reinen Betrieb unter OS/2 sollte man die Medien mit HPFS formatieren. Damit ist ein zügiges Arbeiten der Laufwerke gewährleistet. Möchte man hingegen einige Daten auch mit Nicht-OS/2-Rechnern austauschen, so formatiere man

formatiere man sich zu diesem Zweck einige Zip-Medien mit FAT. Für den ein oder anderen Kopiervorgang ist FAT nicht störend und dürfte nicht allzu sehr aufhalten.

Das *Zip parallel port* ist all jenen zu empfehlen, die ein leichtes, transportables Gerät benötigen. Da es an jeden Parallelport angeschlossen werden kann, bleibt man flexibel. Treiber sind verfügbar und die Installation ist einfach. Die Leistung ist mit HPFS überraschend gut, und da mit einigen Tricks das Wechseln HPFS-formatierter Medien möglich ist, steht einem sinvollen Gebrauch unter OS/2 nichts im Wege. Das *Zip ATAPI* ist eine interessante Erweiterung am IDE-Port. Es zeigt -wieder unter HPFS- gute Leistungen und ist mit Warp 3 und Warp 4, ein entsprechendes Fixpaklevel vorausgesetzt, sehr komfortabel einsatzfähig, besonders was die Benutzung HPFS-formatierter Medien angeht.

Auch der Preis ist akzeptabel: Das externe *Zip 100 parallel port* kostet etwa zwischen DM 250 und 270; *das Zip 100 ATAPI* ca. DM 230-250. Eine 100 MByte-Zip-Diskette bekommt man bereits ab DM 22,00. Gesamturteil: Empfehlenswert! ❑

Iomega Laufwerke und die zum Betrieb nötigen Treiber (* getestet)

Treiber	*Laufwerke*	*Hinweise*
Iomega OAD 2.34	Bernoulli 230	---------
	Multidisk 150	
	Zip 100 SCSI	Unterstützt durch OS/2
	Zip 100 parallel port*	Für den Betrieb mit HPFS-formatierten Medien sind zusätzliche Programme erforderlich
	Jaz 1 GB	Unterstützt durch OS/2
Iomega ADD Filter	Bernoulli 230	---------
	Multidisk 150	---------
	Zip 100 SCSI	Unterstützt durch OS/2
	Jaz 1 GB	Unterstützt durch OS/2
IBM1S506.ADD [1], IBMATAPI.FLT, OS2DASD.DMD (SCSI-ADD)	Zip 100 ATAPI* Zip 100 ATA Zip 100 SCSI Jaz 1 GB (SCSI) Jaz 2 GB (SCSI)	Datenträger können für FAT und HPFS formatiert werden; zum Umgang mit HPFS-formatierten Medien werden keinerlei Zusatzprogramme benötigt. Zum Betrieb der anderen Iomega SCSI-Wechsellaufwerke sind die herstellereigenen Treiber nicht erforderlich.

[1] Neue Version enthalten in Warp 3 ab FixPak 35 (in Warp 4 ab FixPack 6)

Für die voregstellten Iomega-Laufwerke benötigte Software

Iomega Zip 100 parallel port

Treiber: Vom Hersteller verfügbar auf **http://www.iomega.com** oder
http://www.cefischer.de/os2only/software/treiber.htm
Dateiname: **os2v234.exe** (Selbstentpackende DOS-Datei)
Benötigter Treiber aus dem Paket: **Iomega OAD-Treiber**.

Software: Vom Hersteller **LOCK.EXE** zum Sperren/ Entsperren von Laufwerken und dem Ausgeben von Medien

Parameter: lock <laufwerk:> /U - unlock
Parameter: lock <laufwerk:> /L - lock
Parameter: lock <laufwerk:> /Q - query lock state
Parameter: lock <laufwerk:> /E - eject

Vom Hersteller **GENOAD.EXE** zum Auffinden installierter Laufwerke und zur OAD-Treiberkonfiguration.

Alle weiteren in den Iomega Tools enthaltenen Programme werden zum Betrieb des Laufwerkes nicht benötigt.

hpfsrem.exe - zum Umgang mit HPFS-formatierten Medien,
verfügbar mit Auswerfen. CMD. auf:
http://www.cefischer.de/os2only/software/system.htm
Dateien: **hpfsrem.zip**, **auswurf.zip**

Iomega Zip 100 ATAPI

Treiber: Enthalten in OS/2 mit FixPak 36 für Warp 3 (FixPak 6 für Warp 4) oder höher.

Software: Bei Verwendung von FP 36 für Warp 3 bzw. FP 6 für Warp 4 ist ein Fix erforderlich. Verfügbar auf :
http://www.cefischer.de/os2only/software/system.htm
Datei: **ibmatapi.zip**
Ansonsten ist zum Betrieb keine Software erforderlich.





oder willst Du etwa ernsthaft, daß sich an dieser Situation was ändert?

Ein Blick in die X-Akten

Die **WPS** ist schon ein Markenzeichen für OS/2. Eingeführt mit der Version 2.0 bietet sie eine vielseitig konfigurier- und benutzbare Oberfläche, auf der sich auch ungeübte Benutzer rasch zurechtfinden. Die WPS ist jedoch mehr als nur eine objektorientierte Oberfläche, sie basiert auch auf einem objektorientierten Applikationskonzept, dem **SOM (System Object Model)**. Gemäß dem SOM besteht die WPS aus mehreren Klassen, wobei jede Klasse eine bestimmte Aufgabe hat, z. B. gibt es eine Klasse für WPS-Ordner, für das Launchpad usw. Wie in der objektorientierten Programmierung üblich, kann man neue Klassen schreiben, welche auf den alten basieren, ihre Funktionalität übernehmen und neue Funktionen hinzufügen. Auf diese Weise läßt sich die WPS von Programmieren in nahezu unbegrenztem Umfang erweitern, so daß man einige Unanehmlichkeiten ausgleichen kann. Ulrich Möller, ein Freeware-Programmierer, hat's getan. *XFolder* heißt sein Projekt, das in der Version 0.83 vorliegt. Wir wollten wissen, was es kann, und wie es die Arbeit mit der WPS verändert.

Wo gibt's XFolder?

Die stets aktuelle Version von XFolder steht zum Download auf der Homepage des Entwicklers bereit. Auf der Seite: *http://www2.rz.hu-berlin.de/~h0444vnd/xfldr.htm* findet man Links zum Herunterladen. Wir haben außerdem das interessante Programmpaket auf unsere Downloadseite aufgenommen.

Sie benötigen für die *Xfolder*-Version 0.83 außerdem ein **NLS (National Language Support)**-Paket zur Sprachunterstützung. Auf der Homepage des Autors finden Sie entsprechende Links zum Download. Das deutsche NLS-Paket haben wir ebenfalls auf unsere Downloadseiten aufgenommen. Für Versionen vor 0.82 ist der NLS-Support im Programm selbst enthalten.

Installation

Die Installation von *XFolder* ist einfach. Gehen Sie dazu wie folgt vor:

❶ Kopieren Sie nach dem Download die Zip-Dateien am besten in ein Verzeichnis namens XFOLDER auf dem Installationslaufwerk und entpacken Sie sie dort.

❷ Mit der WPS öffnen Sie diesen Ordner und starten das Programm INSTALL.CMD durch einen Doppelklick auf das Objekt. Über die Kommandozeile wechseln Sie in das XFOLDER-Verzeichnis und geben INSTALL ein. Das Installationsprogramm ist selbsterklärend, so daß hier nicht weiter darauf eingegangen werden muß.

❸ Das Installationsprogramm fragt zum Schluß, ob die WPS neu gestartet werden soll. Man quittiert dies mit *Y* und schließt das OS/2-Fenster durch die Eingabe von *exit*.

Nach der Installation von XFolder findet man auf der Arbeitsoberfläche einen Ordner namens *XFolder-Installation*.



Abb. 1: Das Kontextmenü der Arbeitsoberfläche nach der Installation von XFolder

Außerdem haben sich die Ordnersymbole geändert. Sie sehen aus, wie die von Warp 4 und besitzen ein kleines *x* in der linken unteren Ecke. Beim Starten der WPS wird auch das *XFolder-Logo* angezeigt, das in der Mitte des Bildschirms dargestellt wird und sich langsam vergrößert. Wem das nicht gefällt: Man kann das Anzeigen des Logos über die zweite Seite im *Workplace Shell*-Einstellungsnotizbuch auch abschalten, indem man die Option *Animation beim WPS-Start* deaktiviert.

Zur Konfiguration findet man im Ordner *Systemkonfiguration* zwei neue Objekte: *Workplace Shell* und *OS/2 Kernel*. Über die Objekte erreicht man Notizbücher, mit denen sich Einstellungen der WPS und Systemeinstellungen vornehmen lassen. Unter *OS/2 Kernel* verbergen die Einstellungen des Objektes System und erweiterte Einstellungsmöglichkeiten, etwa zu den Dateisystemen, der WPS und dem OS/2-Scheduler. Einstellungen auf den neuen Seiten des Notizbuches wirken sich auf die CONFIG.SYS aus, so daß man zur Veränderung einiger Systemoptionen (z.B. die Anzahl möglicher Threads, Einstellungen des Dateisystemcaches usw.) die CONFIG.SYS nicht extra zum Editieren öffnen muß, sondern die Veränderungen über dieses Notizbuch vornehmen kann. Im *Workplace Shell*-Notizbuch findet man alle Optionen zur Konfiguration von *XFolder*.

Außerdem verfügen alle Ordner im System in ihrem Einstellungsnotizbüchern über die neue Seite *XFolder*, über welche die globalen Einstellungen, die man im *Workplace Shell*-Notizbuch oder wie gehabt über die das Einstellungen-Notizbuch der WPS vornimmt, für jeden Ordner indivduell überschrieben werden können. Außerdem finden sich zahlreiche neue Funktionen in den Kontextmenüs aller Ordner. Wie das Menü der WPS nach der Installation von *XFolder* aussieht, zeigt die Abbildung 1.

Die neuen Funktionen im Kontextmenü

Die Funktionalität von *XFolder* ist vielseitig. Glückerlicherweise verfügt das Programm über eine sehr gute Online-Dokumentation. Bereits auf den ersten Blick aber springen neue Funtkionen ins Auge, die man sicher gleich ausprobiert:

1. *Befehlszeilen*

   Über diesen Menüpunkt kann man verschiedene Kommandozeilen öffnen, standardmäßig DOS- und OS/2-Fenster oder Vollbildschirme. Besonders praktisch ist, daß beim Öffnen gleich in das Verzeichnis des Ordners gewechselt wird, von dem aus man diese Funktion aufgerufen hat. Das erspart ein müsames Aneinanderreihen von *cd*-Befehlen, bis man endlich im gewünschten Verzeichnis ist. Gerade bei sehr langen Verzeichnisnamen eine wohltuende Funtkion.

2. *Neu erstellen*

   Häufig genutzte Objekte können von jedem Ordner aus über diesen Menüpunkt erstellt werden. Die Objekte werden dann in dem Ordner erzeugt, den man gerade benutzt. Ein weiterer Vorteil ist, daß das Kontextmenü des neuen Objektes gleich zum Editieren geöffnet wird, was einen schnellen und umkomplizierten Umgang beim Erstellen neuer Objekte ermöglich und der objektorientierten Arbeitsweise beiträgt. Ein Öffnen des Schablonen-Ordners entfällt damit (das Öffnen dauert ohnehin zu lange, und die Handhabun mit diesem Ordner ist zu umständlich, so daß es einfacher ist, ein neues Objekt mit einer *Speichern unter*-Funktion eines Programms zu erstellen, was der objektorientierten Arbeitsweise natürlich widerspricht. Die neue Funktion ermöglicht also ein konsequent objektorientiertes Arbeiten).

3. *Ordnerinhalt*

   Ebenfalls ein überaus nutzlicher neuer Menüpunkt. Wählt man ihn aus, wird der gesamte Inhalt des Ordners angezeigt. Unterordner in diesem Ordner präsentieren sich als neue Untermenüs. Auf diese Weise erhält man sehr schnell einen guten Überblick bezüglich des Ordnerinhaltes und kann einzelne Objekte gezielt auswählen, ohne zunächst den Ordner zu öffnen und sich durch seinen Inhalt zu quälen. Besonders bei der WPS ist dieser Punkt praktisch, da man alle auf ihr befindlichen Ordner (u.a. auch den Ordner Systemkonfiguration) schnell erreicht.

4. *Lieblingsordner*

   In den Einstellungen aller Ordner findet sich auf der Seite *XFolder* die Checkbox *Lieblingsordner*. Ist sie aktiviert, erscheint in den Kontextmenüs aller Ordner unter dem Menüpunkt *Ordnerinhalt* der Name dieses Lieblingsordners, dessen Inhalt man sich über eine Auswahl des Menüpunktes wie mit der Funktion *Ordnerinhalt* anzeigen lassen kann. Das ermöglicht einen raschen Zugriff auf häufig benötige Ordner mit ihren Objekten von jedem Ordner des Systems und damit auch von WPS aus. Aktiviert man die Funktion *Lieb-*

lingsordner für alle wichtigen Ordner, kann man auf seine gesamte Datenstruktur von der WPS aus bequem zugreifen (Abb. 3).

5. *Dateinamen kopieren*

Mit diesem Menüpunkt läßt sich der Name des Ordners bzw. der slektierten Objekte in einem Ordner in die Zwischenablage übernehmen. Hält man bei der Auswahl des Menüpunktes die Shift-Taste gedrückt, kopiert *XFolder* die vollständige Pfadangabe des Ordners bzw. der ausgewählten Objekte darin in die Zwischenablage. Das kann praktisch sein, wenn man Dateinamen oder vollständige Pfadangaben in Texte importieren oder in OS/2-Fenster übernehmen möchte. In beiden Fällen spart man sich ein lästiges Abtippen.

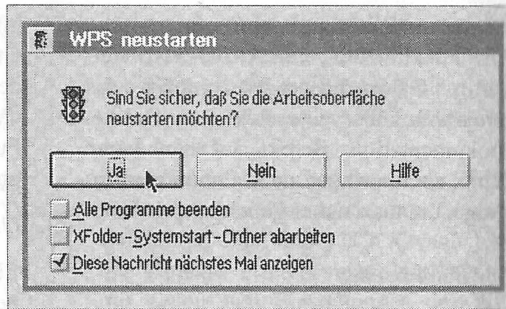


Abb. 2: Der Dialog zum Neustart der WPS

6. *WPS-Neustart*

Im Kontextmenü der Arbeitsoberfläche befindet sich der Menüpunkt *WPS neu starten...* Wählt man ihn aus, wird der Prozeß PMSHELL.EXE



Abb. 3: Der Lieblingsorder Daten im Menü der Arbeitsoberfläche

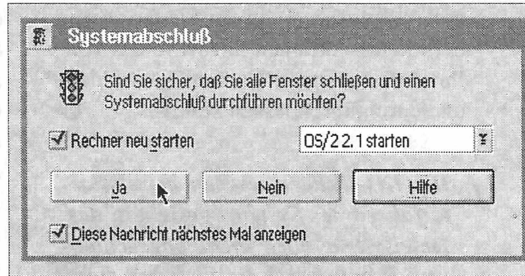


Abb. 4: Der Xfolder-Dialog für den Systemabschluß

durch *XFolder* beendet. Das kommt in etwa dem Fall eines WPS-Absturzes gleich. OS/2 startet anschließend die PMSHELL neu (wie übrigens *jede* Anwendung, die als Arbeitsoberfläche durch die Variable RUNWORKPLACE in der CONFIG.SYS angegeben ist). *XFolder* sichert jedoch alle nötigen WPS-relevanten Daten, bevor er die WPS beendet. Bei uns funktionierte diese Funktion vorzüglich (auch mehrere Male hintereinander) und bietet eine gute Möglichkeit, die WPS bei Störungen und dergleichen auf eine ordentliche Weise zu beenden, um anschließend noch einmal von vorne anzufangen. Ausprobieren sollte man diese Funktion also auf jeden Fall. Sollte es zu Problemen kommen, hilft ein Rechnerneustart, und man kann anschließend die Funktion im Einstellungsnotizbuch *Workplace Shell* deaktivieren. Vor dem Neustart der WPS zeigt *XFolder* ein kleines Dialogfenster an (s. Abb. 2).

7. *Erweiterter Systemabschluß*

Im Kontextmenü der Arbeitsoberfläche kann man auch die Einstellung *Systemabschluß ersetzen* aktivieren. Im Falle eines Systemabschlusses zeigt *XFolder* dann ein Dialogfenster an, das interessante Optionen bietet (s. Abb. 4). So kann der Rechner nach einem Systemabschluß gleich neugestartet werden. Über den Button *Aktionen* des Einstellungsnotizbuches kann man dazu entsprechende Eintragungen vornehmen.

Um zum Beispiel bei installiertem Bootmanager nach dem Systemabschluß den Rechner direkt mit einem anderen Betriebssystem zu starten, gehen Sie wie folgt vor (s. Abb. 5):

❶ Öffnen Sie die Einstellungen der Arbeitsoberfläche. Aktivieren Sie die Option *Systemabschluß ersetzen* und drücken Sie auf *Aktionen*.

❷ Im erscheinenden Dialog klicken Sie auf *Neu* und geben im Feld *Aktionsbeschreibung* einen aussagekräftigen Namen an (z.B. OS/2 2.1 starten).

❸ In der darunterliegenden Kommandozeile geben Sie einen entsprechenden Befehl ein (z.B. setboot /iba:"OS/2 2.1")

❹ Drücken Sie nach den entsprechenden Eingaben auf *OK* und schließen Sie das Notizbuch. Wenn Sie nun einen Systemabschluß durchführen wollen, erscheint zuvor eine kleine Dialog-

box (siehe Abb. 4), in der Sie die zuvor definierten Aktionen mit Hilfe des von Ihnen festgelegten Namens auswählen und durchführen lassen können.

! ***Die INI-Dateien werden beim Runterfahren des Systems gesichert, das Dateisystem aber nicht abgeschlossen. Programme, die Sie über diese Funktion aufrufen, müssen also für derartige Aufräumarbeiten sorgen. Und: Steht das aufzurufende Programm nicht im PATH, muß der vollständige Dateiname angegeben werden.***

! ***Der Treiber DOS.SYS muß installiert sein, um den Rechner nach einem Systemabschluß über die beschriebene XFolder-Option neu zu starten. Überprüfen Sie Ihre CONFIG.SYS, ob Sie eine entsprechende Eintragung finden. Wenn nicht, fügen Sie den Eintrag DEVICE=<Installationslaufwerk>:\OS2\BOOT\DOS.SYS hinzu.***

Beim erweiterten Systemabschluß wird ein Statusfenster angezeigt, über das man Programme, die sich nicht schließen lassen, überspringen, bzw. den Systemabschluß ganz abbrechen kann. Außerdem wird der Fortschritt des Systemabschlusses angezeigt.

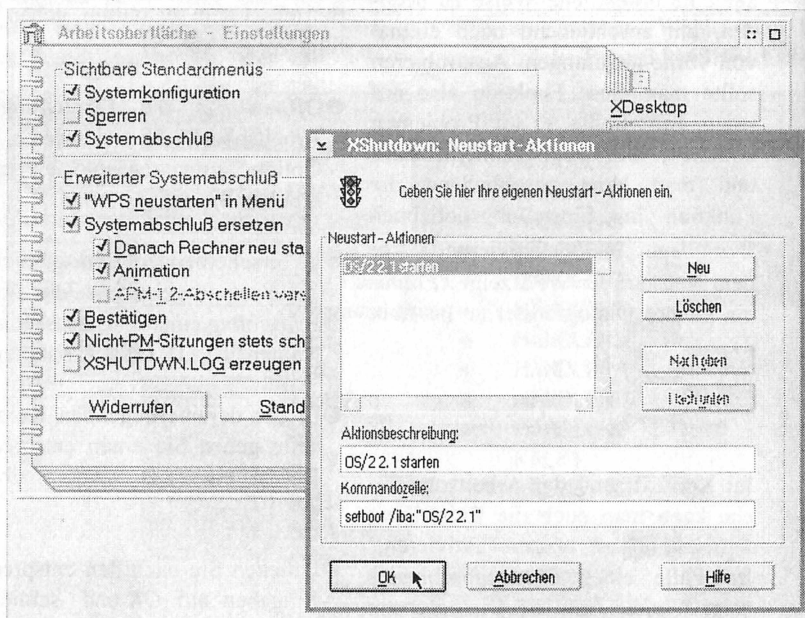


Abb. 5: Das Definieren von Aktionen, die durch XFolder beim Systemabschluß ausgeführt werden sollen

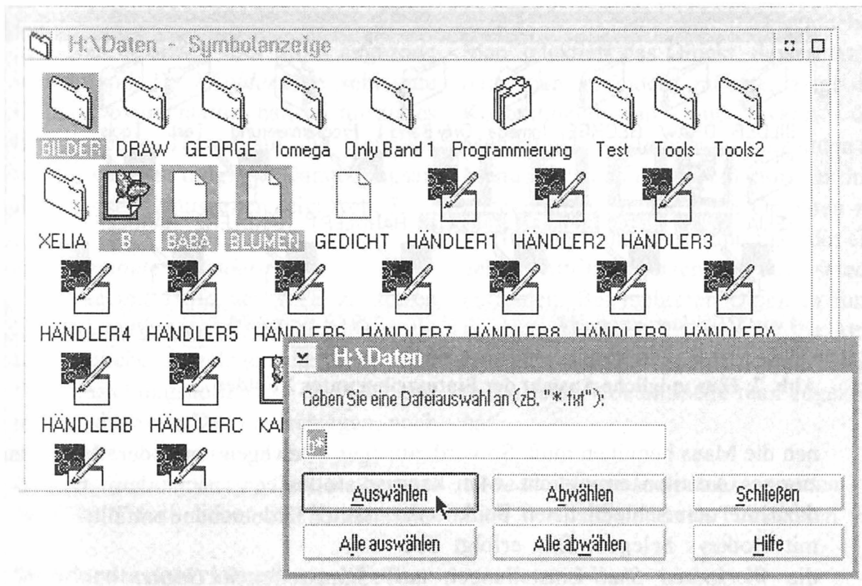


Abb. 6: Die Objektauswahl von XFolder, welche für jeden Ordner zur Verfügung steht.

Sortieren und Auswählen
Es gibt natürlich noch mehr: Eine weitere Besonderheit sind die Sortierfunktionen, die im übrigen global für jeden Ordner über das Einstellungsnotizbuch der Arbeitsoberfläche (Seite *Sortieren*) oder individuell für jeden Ordner festlegbar sind. Über das *Workplace-Shell*-Notizbuch (auch Seite *Sortieren*) kann man die neuen Sortierfunktionen ebenfalls einstellen. Man kann dabei alle bereits bekannten Sortierattribute verwenden, außerdem ein neues namens *Ordner zuerst*, wobei zuerst alle Ordner und dann die Objekte jeweils in alphapetischer Reihenfolge dargestellt werden.
Ein weiteres Plus ist die Funktion *Auswählen nach Namen*: Im *Worplace Shell*-Notizbuch unter Kontextmenüs kann man die Option *Auswählen nach Name* aktivieren. Die Funktion erscheint dann als Menüpunkt im Kontextmenü eines jeden Ordners unter *Auswählen* (Warp 3) bzw. *Anzeigen* (Warp 4). Außerdem definiert *XFolder* das Tastaturküzel *STRG+S* für diese Funktion. Wählt man diesen Menüpunkt aus, erscheint eine Dialogbox, in der man ein Auswahlkriterium angeben kann (z.B. *.c oder b*). Mit einem Klick auf *Auswählen* werden die den Kriterien entsprechenden Objekte selektiert. Eine Abwahl oder die Auswahl aller Objekte ist ebenso möglich (Abb. 6).

Tastaturkürzel
XFolder erlaubt das Verwenden von Tastaturkürzeln, was von großem Nutzen ist, da man nicht für jede Ordnerfunktio-

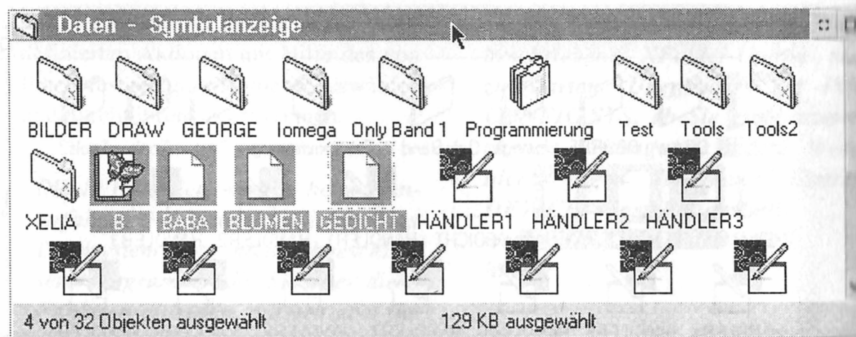


Abb. 7: Eine mögliche Ansicht der Statuszeilen unter XFolder

nen die Maus bemühen muß. So wird ein zügiges Arbeiten ermöglicht. Man kann dazu die unterschiedlichsten Funktionen mit Hotkeys belegen. Das erfolgt über die *Workplace Shell*-Einstellungen auf der Seite *Ordner-Tastaturkürzel*. *XFolder* definiert eine Reihe von Standardtastaturkürzeln, die man auch überschreiben kann. Das sollte man nicht unbedingt tun, denn sie sind allesamt recht nützlich. Eine Auflistung aller Standardkürzel finden Sie daher im Kasten *XFolder Standardtastaturkürzel*. Neben den Computer gelegt, hilft diese Liste beim Lernen der Kürzel. Einstellungen, die man bezüglich der Hotkeys vornimmt, werden global für alle Ordner aktiv.

Automatisches Scrollen im Treeview

XFolder ermöglicht das automatische Scrollen von Strukturanzeigen. Bei dieser Funktion werden Strukturanzeigen beim Öffnen einer weiteren Ordnerebene automatisch nach unten gescrollt, so daß die neuen Unterordner sichtbar werden. Damit wird die Arbeit mit der Strukturanzeige komfortabler, weil ein ständiges Bewegen mit der Maus zu den Scrollbalken nach dem Öffnen einer neuen Ordnerebene entfällt.

Statuszeilen für Ordner

Statuszeilen dürfen für Ordner nicht fehlen. *XFolder* bietet auch Statuszeilen, die sogar sehr flexibel sind und zwar gleich in drei verschiedenen Looks. Die Anzeige der Informationen erfolgt ebenfalls in drei unterschiedlichen Arten: Ist kein Objekt selektiert, wird die Gesamtzahl der Objekte im Ordner angezeigt; ist nur ein Objekt ausgewählt, werden dessen Titel, Dateityp und Dateigröße angezeigt. Wenn mehrere Objekte selektiert sind, wird die Anzahl der selektierten Objekte, die Gesamtgröße der selektierten Objekte und die Gesamtan zahl der Objekte im Ordner dargestellt. Farben und Schriftarten können aus den Paletten im *Systemkonfigurations*-Ordner auf die Statustzeilen gezogen werden, so daß man sich deren Aussehen nach eigenem Gefallen mühelos gestalten kann. Derartige Änderungen werden übrigens auch in diesem Falle global aktiv. Mit Hilfe

von Schlüsseln kann man genau festlegen, was in diesen drei Modi angezeigt werden soll. Da *XFolder* eine sehr gute Online-Dokumentation hat, sei für dieses tiefergehende Thema auf die Referenz verwiesen. Eine mögliche Darstellungsart der *XFolder*-Statuszeilen zeigt Abb. 7.

Dateiatrribute verändern
Attribute mit Hilfe der WPS zu setzen und zu entfernen ist schon eine recht umständliche Angelegenheit: Man muß das Einstellungsnotizbuch des Objektes öffnen, die Seite Datei aufschlagen, noch eine Seite nach vorne blättern und kann dann endlich ans Werk gehen. Da benutzt man dann doch lieber die Kommandozeile. Mit *XFolder* geht das einfacher: Man selektiert das Objekt, dessen Attribute man verändern möchte, zeigt das Kontextmenü an und wählt den Menüpunkt *Dateiattribute*, ganz unten im Menü, der nach der Installation neu hinzugekommen ist und allen Objekten zur Verfügung steht. Dort kann man die einzelnen Attribute setzen und auch wieder entfernen. Bei mehreren Objekten funktioniert das aber nicht, weil nur die Attribute des Objektes geändert werden können, dessen Kontextmenü man angezeigt hat.

Neue Objekte unkompliziert erstellen
Eine der bedeutendsten Funktionen von

Standardtastaturkürzel von XFolder

Tastaturkürzel	***Funktion***
Backspace	Stammordner öffnen
Shift+Backspace	Stammordner öffnen und aktiven Ordner schließen
F5	Ordnerinhalt sofort aktualisieren
Strg+A	Alle Objekte selektieren
Strg+D	Alle Objekte deselktieren
Strg+E	Objekte nach ihrer Dateierweiterung sortieren
Strg+N	Objekte nach ihrem Namen sortieren
Strg+S	Aufruf der XFolder-Funktion *Nach Name auswählen*
Strg+W	Objekte nach dem Datum letzten Schreibens sortieren
Strg+Y	Objekte nach ihrem Typ sortieren
Strg+Z	Objekte nach ihrer Größe sortieren
Strg+Einfügen	Dateinamen der selektierten Objekte in die Zwischenablage kopieren
Strg+Shift+D	Detailanzeige des Ordners öffnen
Strg+Shift+I	Symbolanzeige des Ordners öffnen
Strg+Shift+S	Einstellungsnotizbuch des Ordners öffnen

XFolder ist die Möglichkeit, über die Kontextmenüs eines jeden Ordners neue Objekte zu erstellen. Man wählt dazu den Menüpunkt *Neu erstellen* aus und findet dort weitere Menüpunkte für jedes Objekt, das man neu erzeugen kann. Standardmäßig existieren bereits zwei Menüpunkte: *Ordner* und *Programmobjekt*. Eine Erweiterung des Menüpunktes ist aber einfach möglich. So kann man etwa vorkonfigurierte Schablonen oder Programmobjekte diesem Menü hinzufügen, ohne daß man den Schablonenordner öffnen müßte (was ohnehin zu lange dauert und zu unübersichtlich ist). Die Erweiterung des *Neu erstellen*-Menüpunktes ist dabei einfach. Neue Einträge erfolgen über den *XFolder*-Konfigurationsordner, der sich nach der Installation im Ordner *XFolder-Installation* befindet. Man sollte sich diesen Ordner oder eine Refernz besser zugänglich auf den Desktop oder in den Ordner *System* legen. Öffnet man den Ordner, findet man weitere Ordner darin vor, deren Namen man schon von den Punkten in den Kontextmenüs kennt. Wie man sieht, kann man also seine Ordnerkontextmenüs mit Hilfe dieses Ordners rasch erweitern, indem man einen neuen Ordner darin ablegt. Mit Hilfe des ~ Zeichens kann man auch einen Shortcut fest-

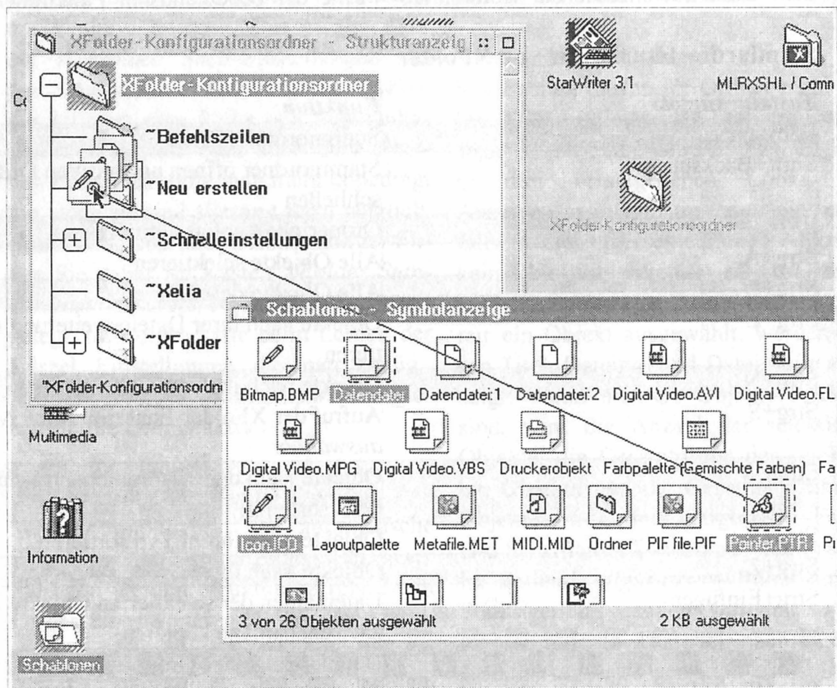


Abb. 8: Menüpunkte zu den Kontextmenüs der Ordner hinzufügen. Hier werden Einträge für den Menüpunkt Neu erstellen erzeugt.

legen. So ist die Möglichkeit, Kontextmenü selbst zu erweitern, völlig selbsterklärend: Untermenüs sind Ordner; Menüpunkte Objekte in diesen Ordnern. Um also eine Schablone als Eintrag in den *Neu erstellen*-Menüpunkt hinzuzufügen, brauchen Sie nur den *Konfigurationsordner* öffnen, und eine eine Referenz der gewünschten Schablone aus dem Schablonenordner in den Unterorder *~Neu erstellen* des *Konfigurationsordners* ziehen.

Damit ist die Arbeit getan. Nun findet man in dem Menüpunkt *Neu erstellen* eines jeden Ordners die Schablone als entsprechenden Eintrag wieder. Wählt man den Eintrag aus, wird in dem Ordner, mit dem man gerade arbeitet, das neue Objekt erstellt. Wie das Ganze in Aktion aussieht, zeigt Ihnen Abb. 8.

Was bleibt?

Ein guter Eindruck und jede Menge andere neue Einstellungen und Funktionen, die wir im Rahmen dieses Artikels jedoch nicht beschreiben konnten. *XFolder* ist eine gut durchdachte Roßkur für die WPS, die vor allem nicht die Ressourcen des Systems unangenehm in Anspruch nimmt und - *XFolder* ist Freeware!

Enige sehr störende Lücken, die IBM in der objektorientierten Benutzerführung der WPS offenließ, werden sauber geschlossen, die Arbeit mit der Benutzeroberfläche enorm erleichtert, und auch die Dokumentation ist verständlich und umfassend, womit einem genauen Kennenlernen des Programms nichts im Wege stehen sollte. *XFolder* läuft (obgleich als Beta) stabil und verursachte auf unseren Rechnern keine Probleme. Bis auf einige Probleme mit ObjectDesktop, auf die der Autor hinweist, arbeitet es auch mit anderen WPS-Erweiterungen gut zusammen, etwa mit NPS-WPS. Wer diese Programme einsetzt, kann *XFolder* also ebenfalls ohne Schwierigkeiten ausprobieren. Dennoch sollte man sich die Hinweise in der Online-Referenz, welche die Kompatibilität mit anderen Programmen betreffen, vor dem Ausprobieren sorgsam durchlesen. Überhaupt sind Probleme nur mit solchen Programmen wahrscheinlich, die ebenfalls Manipulationen an der Klassenstruktur der WPS durchführen.

Sollte mal etwas schiefgehen, erstellt *XFolder* eine LOG-Datei mit Debug-Informationen, die man an den Autor schicken kann, um ihm bei der Fehlersuche behilflich zu sein. Daneben läßt sich *XFolder* sehr einfach deinstallieren, sollte man nach dem Test des Programms doch nicht überzeugt sein. Zuvor sollte man aber, wie im übrigen vor der Installation, die WPS sichern, was sich über die Archivierungsfunktion von OS/2 einfach bewerkstelligen läßt.

Wer gerne mit der WPS arbeitet, sollte sich *XFolder* also mal etwas genauer ansehen. Ein Blick in den X-Ordner dürfte sich lohnen. ❑

Werkzeuge für das System

Man braucht für alltägliche Aufgaben immer wieder ganz bestimmte Programme, sei es zur Pflege der Datenbestände, zum Testen des Computers oder ähnlichem. Wir führen daher einmal in diesem Artikel einige interessante Werkzeuge auf, die man eigentlich immer braucht. Besonders all diejenigen unter Ihnen, die auf OS/2 umsteigen, werden hier vielleicht einen kleinen Leitfaden finden.

Programme für die Systemdateien

In diesem Abschnitt finden Sie Programme zur Verwaltung der Systemdateien, also der INI-Dateien und der CONFIG.SYS.

1. Die CONFIG.SYS sortieren

Die CONFIG.SYS ist nach der Installation von Warp ein einziges Durcheinander. Warum man Sie in einen ordentlichen Zustand überführen sollte, erklären wir Ihnen im Kapitel *Know how*, wo wir auch zeigen, wie man es von Hand macht. Es gibt aber auch ein kleines Werkzeug namens *CFGSORT*, welches das Sortieren einer CONFIG.SYS übernimmt. Sie finden es auf unserer Homepage unter *cfgsrt22.zip*. Das Programm arbeitet Sowohl unter Warp 3 als auch unter Warp 4 zuverlässig. Es ist ein compiliertes REXX-Programm, und benötigt zur Ausführung eine spezielle DLL, die jedoch mitgeliefert wird. Da das Programm zuverlässig ist und einem eine Menge Arbeit abnimmt, hier die Anweisungen zur Installation und zum Gebrauch:

❶ Entpacken Sie die ZIP-Datei in ein Verzeichnis, das in der PATH-Anwei sung steht oder erzeugen Sie ein Ver zeichnis, z.B. \CFGSRT, das Sie zum PATH hinzufügen.

❷ Tragen Sie die Umgebungsvariable

SET CONFIGSORT=<Instl.-Verz.>

in die CONFIG.SYS ein, wobei Instl Verzeichnis die vollständige Pfadan gabe des Verzeichnisses ist, in dem Sie CFGSRT installiert haben.

❸ Fügen Sie den Pfad:

<Instl.-Laufw.>:\<Instl.-Verz.>\DLL

der LIBPATH-Anweisung in der CONFIG.SYS hinzu, wobei *Instl.-Laufw.:\<Instl.-Verz.>* der vollständi gen Pfad des Installationsverzeichnis ses von *CFGSRT* ist.

❹ Führen Sie einen Systemabschluß durch und starten Sie den Rechner neu.

CFGSRT wird über die Kommandozeile bedient. Der Gebrauch ist einfach. Um etwa die Datei c:\config.sys zu sortieren, gibt man ein:

cfgsrt /in c:\config.sys /out c:\config.sys

Dabei wird die alte Datei gleich durch die neue ersetzt.

! ***Der Autor gibt an, der /in-Parameter sei die einize obligatorische Angabe, und die alte Datei werde***

als CONFIG.BAK gesichert. Auf unseren Systemen hängte sich das Programm jedoch auf, wenn man den /out-Parameter wegließ.

Hinter /out kann man jede Angabe machen, etwa config.new. So wird die alte config.sys nicht gleich ersetzt.

CFGSRT unterteilt die Datei in Sektionen, wie *Base Device Drivers*, Installable File Systems usw. Alle SET-Anweisungen werden alphabetisch geordnet. Übrigens werden alle Einträge bis auf diejenigen, welche mit SET beginnen, in Großbuchstaben umgewandelt. Das ist für die CONFIG.SYS auch angebrachter. Wer das nicht will: Diese Funktion läßt sich abschalten. Eine Zusammenstellung aller Parameter finden Sie im Kasten Parameter von CFGSORT.

Parameter von CFGSRT

/in <Quelldatei>
vollständiger Pfad der CONFIG.SYS

/out <Zieldatei>
vollständiger Pfad der Zieldatei

/appcfg <Konfigurationsdatei>
vollständiger Pfad einer Konfigurationsdatei für *CFGSRT*

/noapps
mehrfache Einträge eines Programms in der CONFIG.SYS werden in einer eigenen Sektion am Ende der Datei zusammengefaßt. Der Parameter deaktiert diese Funktion.

/nosort
SET-Anweisungen werden von *CFGSRT* alphabetisch geordnet. Der Parameter deaktiviert diese Funktion.

/noupper
Bis auf die Eintragungen, die mit dem Schlüsselwort SET eingeleitet werden, wandelt das Programm alle weiteren Eintragungen in der CONFIG.SYS in Großbuchstaben um. Der Parameter deaktiviert diese Funktion.

2. INI-Dateien verwalten

Bekanntermaßen sammeln sich v.a. in der OS2.INI mit der Zeit unnötige Einträge, welche die Datei aufblähen und die Verwaltung verlangsamen. Das liegt daran, daß leider nach wie vor viele OS/2-Programme die schlechte Angewohnheit haben, ihre Einstellungen in der OS2.INI festzuhalten, anstatt eine eigene INI-Datei zu erzeugen und zu verwalten. Wenn man irgendwann einmal eines derartig garstiger Programme löscht, bleibt dessen Eintrag als Relikt in der OS2.INI zurück. Daher sollte man diese Datei von Zeit zu Zeit von unnötigem Ballast befreien.

Dazu gibt es sogenannte INI-Editoren und davon wieder eine ganze Menge. Wir haben uns *MIni* von Kai Evers aus dieser Menge herausgegriffen. Zwar greift es auf REXX zurück und gehört damit vielleicht nicht zu den schnellsten Programmen; aber es besitzt eine angenehme Oberfläche und bietet in seiner Schlichtheit alle Funktionen, die man braucht. Für die regelmäßige Arbeit ist dieses kleine Werkzeug also wie geschaffen. Die Installation ist simpel: Entpacken Sie die ZIP-Datei am besten in ein Verzeichnis, das sich im PATH

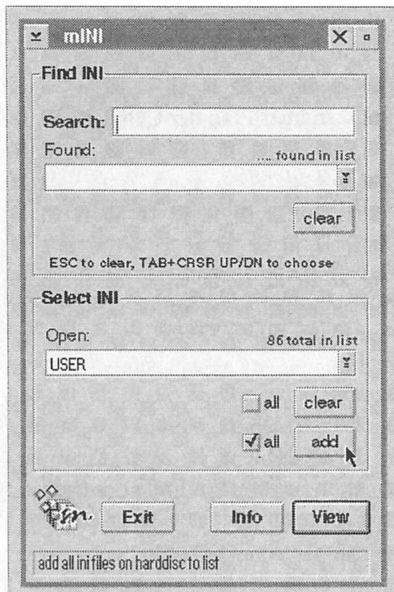


Abb. 1: Das Eingangsfenser von MIni

befindet, damit sie es von der Kommandozeile jederzeit aufrufen können, und -sofern Sie mit der WPS arbeiten- erzeugen Sie ein Programmobjekt, wovon Sie eine Refernz praktischerweise auf das Launchpad oder das WarpCenter legen sollten. Das war's auch schon.

Nachdem *MIni* gestartet ist, präsentiert es sich wie in Abbildung 1. In der *Select INI Group* des Programmfensters kann man über eine Listbox mit den Eintragungen USER die OS2.INI und SYSTEM die OS2SYS.INI erreichen. Versieht man das Feld all mit einem Häkchen und drückt auf add, wird das gesam System nach INI-Dateien durchsuch die nach dem Suchvorgang in d Listbox zur Auswahl bereitstehen. Ma wählt dann einfach die gewünschte Dat aus und drückt auf *View*, wodurch s geöffnet wird. Der Inhalt der Datei pr sentiert sich dann wie in Abb.2 in eine übersichtlichen Baum, der die Auswa aller in der INI-Datei gespeicherten Ei träge (Applications) mit ihren spezi schen Schlüsseln (Keys) bietet. M kann einzelne Keys öffnen und editiere und sowohl sie als auch den ganzen Ei trag löschen. Nach dem Editier schließt man die *View*-Box mittels *Clo* und *MIni* durch Drücken auf *Exit*. Dar werden die Veränderungen auch gespe chert.

Aber Vorsicht: Man sollte nur die Ei träge löschen, von denen man weiß, d sie unbrauchbar sind. Die Einträge d einzelnen Anwendungen sind durch d

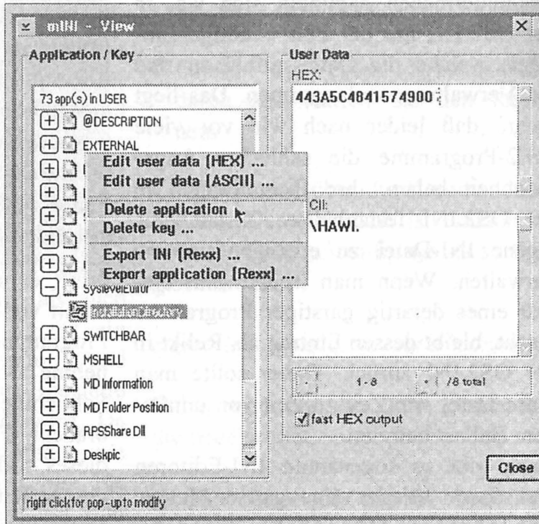


Abb. 2: Das View-Fenster von MIni

Namen des Programms deutlich gekennzeichnet, und da man weiß, welche Anwendungen auf dem eigenen Rechner noch ihren Dienst tun und welche nicht, läßt sich in aller Regel leicht entscheiden, ob man den INI-Eintrag löschen kann. Im Zweifelsfall beläßt man den Eintrag in der INI-Datei.

! ***Für die OS2.INI gilt: Finger weg von allen Einträgen, die mit PM_ beginnen! Außerdem von dem Eintrag SYS_DLLS und Einträgen, die Druckerobjekte betreffen (sie tragen den Namen des Treibers, etwa IBMPCL5).***

Programme für die Datenpflege

In diesem Abschnitt stellen wir Ihnen Programme für den Umgang mit Dateien und Verzeichnissen vor.

1. Dateimanager für den Notfall

Dateimanager können auch unter OS/2 wichtig sein, wenngleich man für den Umgang mit Daten jeglicher Art sicher die WPS bevorzugt. Wenn man aber mit der WPS nicht arbeitet oder einen komfortablen Zugriff auf die Daten eines Systems von einer Wratungspartion oder einer Notfalldiskette haben möchte, kommt man an diesen Programmen nicht vorbei. Es gibt für OS/2 eine Vielzahl von Dateimanagern, einen davon verwenden wir schon seit längerem und wollen ihn daher vorstellen.

Den *File Commander* kennt man unter anderem Namen auch von DOS her. Das OS/2-Pendant, das seinem Original sowohl vom Äußeren als auch seiner Funktion nach gleicht, arbeitet zuverlässig und erlaubt einen komfortablen Zugriff auf alle HPFS- und FAT-Laufwerke. Da das Programm sehr handlich ist, kann man es bequem auf einer Notfalldiskette unterbringen, so daß man im Falle eines Falles nicht auf die Komman - dozeile angewiesen ist, um schnell mit Dateien arbeiten zu können. Aber auch sonst eignet sich das Programm, selbst wenn man die WPS verwendet, da es einen weitaus schnelleren Umgang mit Dateien und Verzeichnissen gestattet, zudem noch bequemer als über den Prompt. Das Programm ist selbsterklärend und wird über die Funktionstasten bedient. Am unteren Bildschirm- oder Fensterrand findet man eine Beschreibung, welche Tasten welche Funktion erfüllen. Wie der *File Commander* aus - sieht, zeigt Abb. 3.

2. Gelöschte Dateien wiederherstellen

Die OS/2-Undelete-Funktion ist nicht unbedingt eine optimale Einrichtung zum Wiederherstellen gelöschter Dateien. Sie arbeitet zwar recht zuverläs - sig, jedoch setzt sie die Systemleistung herab, da Dateien nicht wirklich gelöscht, sondern in ein Verzeichnis verschoben werden, das durch die Umgebungsvariable DELDIR in der CONFIG.SYS festgelegt wird. Daher ist diese Funktion unter OS/2 auch standardmäßig deaktiviert. Und wenn aus dem DELDIR-Verzeichnis eine Datei vom System gelöscht wird, oder man auch Unachtsamkeit wichtige Daten von der Platte entfernt, benötigt man Programme, die trotzdem ein Wiederherstellen ermöglichen. Wir stellen ihnen zwei davon vor.

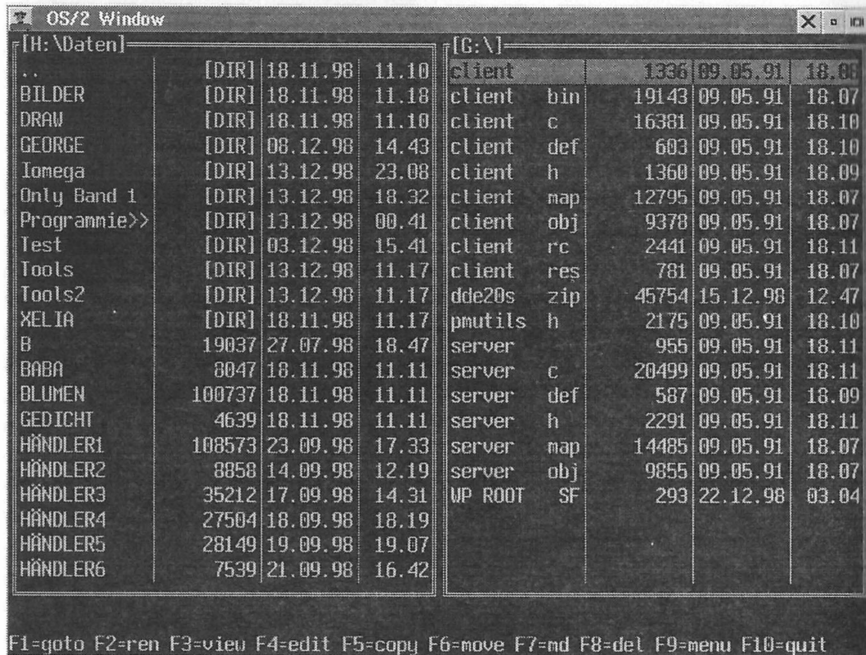


Abb. 3: Der FileCommander als geeignetes Tool für den Umgang mit Dateien als Alternative zum Prompt

File Phoenix ist ein IBM-EWS-Programm und dürfte das bekannteste Datei-Wiederherstellungsprogramm für OS/2 sein. Es liegt mittlerweile in der Version 1.35 vor, unterstützt HPFS wie FAT und erlaubt, im Gegensatz zu früheren Versionen, auch die Rettung von Dateien mit langen Namen (zuvor stellte es nur Dateien wiederher, die nach der 8.3-Konvention benannt waren) und von Dateien, die Leerzeichen in ihrer Bezeichnung haben. Die Installation besteht eigentlich nur aus dem Entpakken der ZIP-Datei in ein Verzeichnis, das am besten wieder im PATH eingetragen ist (\OS2\APPS ist ein guter Ort). Zusätzlich können Sie noch ein Programmobjekt auf der WPS erstellen. Da es sich um ein PM-Programm handelt, ist für den Gebrauch im Notfall eine Wartungspartition vonnöten, die mit dem PM eingerichtet wurde.

Die Bedienung ist leicht: Man wählt ein Laufwerk aus und startet die Suche nach wiederherstellbaren Dateien duch einen Doppelklick auf den Phönix im rechten Teil des Fenster oder mit Hilfe des Menüpunktes *Scan*. Nach dem Scan werden alle wiederherstellbaren Dateien, die das Programm gefunden hat, in einem Container zur Auswahl angezeigt. Mittels eines Kontextmenüs kann man die ausgewählten Dateien wiederherstel-

len (Abb. 4). *File Phoenix* verfügt über eine sehr gute, kontextsensitive Online-Hilfe, so daß wir nicht näher auf das Programm eingehen müssen.

! ***Geben Sie in den Dialogen zur Wiederherstellung immer ein Verzeichnis an, in welches die geretteten Dateien abgelegt werden sollen. Beachten Sie fernerhin, die Dateien nicht auf dem Laufwerk wiederherstellen zu lassen, auf welchem sie gefunden wurden.***

Mit *File Phoenix* lassen sich gelöschte Dateien mit guter Aussicht auf Erfolg wiederherstellen. Wir verwenden das Programm regelmäßig und hatten damit nie Probleme. Aber -und Sie wissen, es gibt immer ein Aber- zögern Sie nicht zu lange mit der Wiederherstellung gelöschter Dateien. Je schneller Sie nach dem Löschen versuchen, sie zu retten, desto größer ist die Wahrscheinlichkeit, sie auch tatsächlich wiederherstellen zu können. File Phoenix ist ein schlichtes Programm ohne jeden Schnörkel, erfüllt aber seinen Dienst ohne Beanstandung.

Wer eine Funktion zur Wiederherstellung gelöschter Dateien auf FAT-Partitionen nicht benötigt, kann auch auf ein Shareware-Programm namens *HPFS Tool 1.7* zurückgreifen, das eine Undelete-Funktion bietet. Dieses Programm ist ebenso einfach zu bedienen wie *File Phoenix* und auch die Installation ist diesselbe.

Nach dem Programmstart wählt man den Menupunkt *Undelete*, wählt im erscheinenden Fenster ein Laufwerk aus und startet den Scan der Partition. Die Benutzerführung ist selbsterklärend. Wie die Oberfläche aussieht, zeigt Abb. 5.

Die Undeletefunktion von *HPFS Tool* ist genauso zuverlässig wie die von *File Phoenix*. Der Unterschied liegt nur darin, daß *HPFS Tool* ein wenig schneller bei der Suche nach gelöschten Dateien zu Werke geht und eine bessere Statusan-

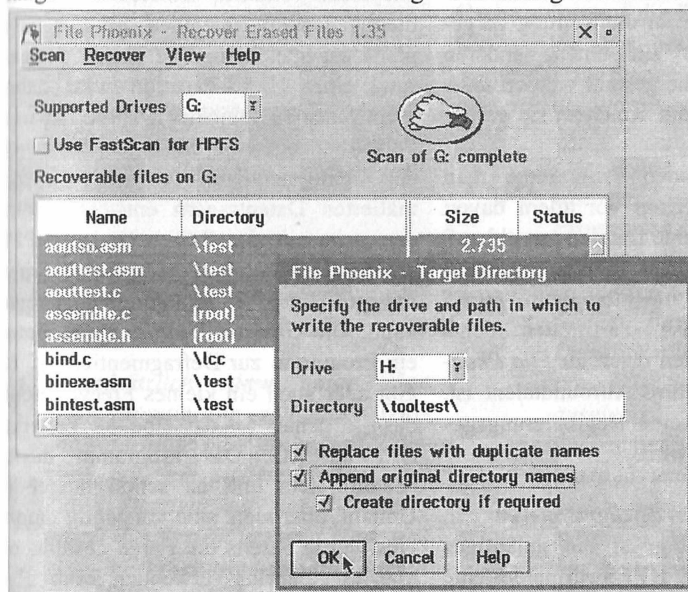


Abb. 4: Der Wiederherstellungsdialog von File Phoenix

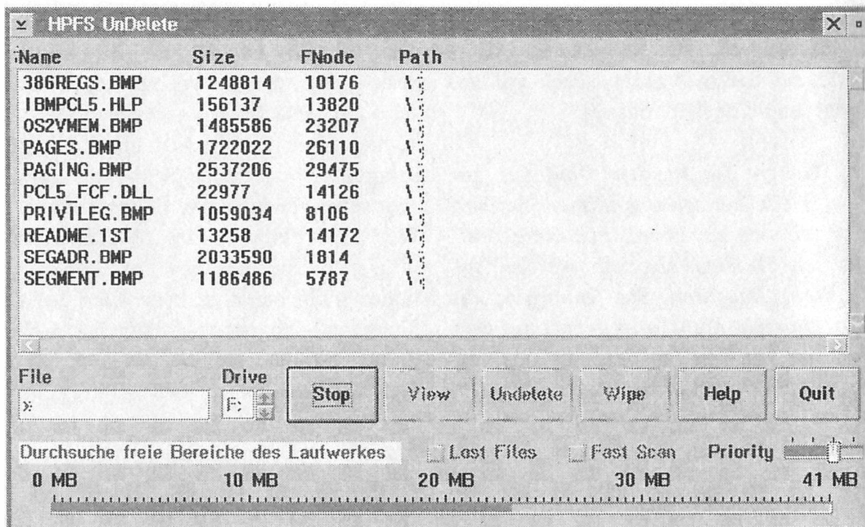


Abb. 5: Die Oberfläche der Undelete-Funktion von HPFS Tool 1.7

zeige besitzt. Zu beachten ist hier aber ebenfalls, daß die für die Rettung vorgesehenen Dateien auf einem anderen Laufwerk wiederhergestellt werden müssen als auf dem, auf welchem sie gefunden wurden.

Welches der beiden Programme man nun verwendet, hängt vor allem davon ab, ob man gelöschte Dateien sowohl auf **FAT**- als auch auf HPFS-Partitionen wiederherstellen möchte, denn *HPFS Tool* kann mit FAT-Partitionen nichts beginnen. Wir raten daher zu *File Phoenix*, da es ein wahres Allroundtalent ist, für das zudem keine Registrierungsgebühr erforderlich wird.

3. HPFS-Partionen defragmentieren

Dateifragmentierung ist ein geläufiges Problem mit FAT. **HPFS** vermeidet eine Fragmentierung fast ganz, da das Dateisystem versucht, Dateien in zusammenhängenden Bereichen auf der Festplatte zu speichern. Die Defragmentierung einer HPFS-Partition wird daher nicht einen so subjektiv wahrnehmbaren Performancegewinn erbringen, der durch das Defragmentieren von mit FAT formatierten Datenträgern entsteht. Trotzdem tritt mit der Zeit auch bei HPFS eine leichte Fragmentierung auf, die man mit geeigneten Programmen beseitigen kann. Die *Graham Utilities* etwa bieten ein Programm zur Defragmentierung. Es gibt aber auch ein kleines Freewaretool, das genau dasselbe bewirkt: *HPPFSDFRG*. Das Programm dürfte bekannt sein und ist schon länger in Umlauf, aber viele sind vorsichtig damit. Uns wurde bereits die Frage gestellt, ob man es gefahrlos verwenden kann, also haben wir es getestet.

Parameter von HPFSDFRG

Syntax: *hpfsdfrg [-sr] <Verzeichnis>*

-s
Alle Unterverzeichnisse des im Befehl angegebenen Verzeichnisses werden mit diesem Parameter ebenfalls defragmentiert. Beispiel:

hpfsdrg -s e:\daten

-r
Dateien, die momentan in Gebrauch sind, werden von hpfsdfrg übersprungen. Beispiel:

hpfsdrg -r e:\daten

Um ein ganzes Laufwerk zu defragmen - tieren, geben Sie ein:

hpfsdfrg -s e:

Um ein ganzes Laufwerk zu defragmen - tieren, dabei aber alle Dateien zu überspringen, die momentan in Gebrauch sind, geben Sie ein:

hpfsdfrg -sr e:

! ***Grundsätzlich nimmt man eine Defragmentierung von einer Wartungspartition oder einer Notfalldiskette vor, um sicherzustellen, daß keine Dateien auf der Festplatte durch das System gesperrt sind, was eine Verarbeitung dieser Dateien ausschließen würde.***

Entpacken Sie die ZIP-Datei und kopieren Sie HPFSDFRG.EXE auf eine Notfalldiskette oder eine Wartungspartition. Nach der Empfehlung des Autors verändern Sie in der CONFIG.SYS auf Diskette 2 der Notfalldisketten oder auf der Wartungspartition die Eintragungen, die das Dateisystem betreffen. Setzen Sie die Größe des Diskcaches auf 64 KByte mit dem Parameter /CACHE:64 in der IFS-Anweisung. Die Anweisung kann dann z.B. so aussehen:

IFS=D:\OS2\HPFS.IFS /CACHE:1024

Fügen Sie außerdem folgende Zeile in die CONFIG.SYS ein:

RUN=CACHE.EXE /LAZY:OFF

Dieser Eintrag bewirkt, daß Daten im Cache sofort auf die Festplatte geschrieben werden. Mit diesen Verändernugen starten Sie Ihr System mit den Disketten oder von der Wartungspartition aus.

Tip: *Der Autor von HPFSDFRG rät, vor dem Gebrauch des Programms ein Backup der zu defragmentierenden Dateien anzufertigen. Man sollte ein solches Back daher vor der Defragmentierung durchführen. Wir haben HPFSDFRG ausgiebig mit Partitionen unterschiedlicher Größen unter Warp 3 und Warp 4 getestet und waren mit dem Ergebnis zufrieden. Schwierigkeiten traten keine auf - aber für den Fall der Fälle sollte man ein*

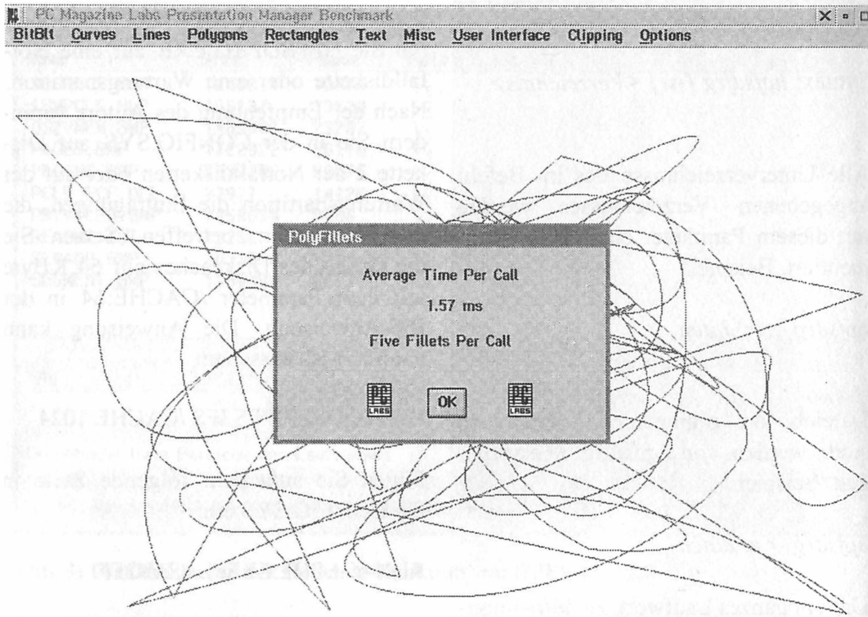


Abb. 6: PMBench nach einem der Graphiktests. Das Ergebnis wird in Millisekungen angegeben

Backup durchführen. Da Backups das A und O in der EDV sind, bietet eine Defragmentierung auch einen guten Anlaß zum Sichern der Daten.

Anschließend kann man *HPFSDFRG* von der Kommandozeile einsetzen. Das Programm ermöglicht die Defragmentierung einzelner Verzeichnisse oder ganzer Partitionen. Es gibt dazu zwei Parameter deren Bedeutung Sie dem Kasten Parameter für *HPFSDFRG* entnehmen können.

HPFSDFRG nutzt die Tatsache, daß **HPFS** Dateien in zusammenhängenden Bereichen auf die Festplatte schreibt. Also kopiert es eine Datei einfach in eine neue Datei ins gleiche Verzeichnis und löscht nach erfolgreichem Kopiervorgang das Original. Anschließend benennt es das zuvor abgespeicherte Duplikat mit dem Namen des Originals. Mehr passiert nicht. Man kann daher den gleichen Effekt erreichen, indem man ein komplettes Backup einer Partition anfertigt, sie formatiert und die Dateien wieder zurückspielt. *HPFSDFRG* ist aber flexibler, weil man gezielt auch nur einzelne Verzeichnisse defragmentieren kann.

Wir fragmentierten unter anderem eine reine Datenpartition. Nach der Defragmentierung ging alles schon ein wenig schneller vor sich. Das mag aber auch daran gelegen haben, daß die Partition

seit über einem Jahr ständig in Gebrauch war und nie defragmentiert wurde. Bei regelmäßig durchgeführten Defragmentiervorgängen dürften Leistungsveränderungen weniger stark auffallen. Leistungsverbesserungen gehen Verschlechterungen voraus. Also sollte man auch eine HPFS-Partition von Zeit zu Zeit ein wenig aufräumen. *HPFSDFRG* ist dafür ein geeignetes und kostenloses Mittel.

Benchmarkprogramme

Der Nutzen von Benchmarks sei einmal dahingestellt. Im wesentlichen kann man sie genau so mißbrauchen wie Statistiken. Allerdings bieten sie einen Bezugspunkt zur Beurteilung der Systemleistung. Das heißt aber nicht, daß man sein System mit der Hilfe von Benchmarks optimieren sollte. Es mag sein, daß es dann eine optimale Leistung zeigt, wenn man das Benchmark programm ausführt; aber die Gesamtleistung des Systems leidet eher darunter. Für eine grobe Beurteilung ist ein Benchmark aber wie gesagt nicht schlecht. Daher stellen wir Ihnen zwei Benchmarkprogramme vor, die für OS/2 verfügbar sind (die Auswahl ist hier nicht so groß) und sich für einen Einsatz eignen. Wir werden für spätere Hardwaretests diese Programme verwenden, damit Sie einen bekannten Bezugspunkt haben.

PMBench

PMBench wurde von *Graphic Software Systems* für *PC Magazin* entwickelt. Das Programm liegt nur in der Version 1.0 vor und ist -bekommen Sie keinen Schreck- schon neun Jahre alt. Jedoch arbeitet es zuverlässig und gibt gute Anhaltspunkte über die graphischen Leistungen des PM. Vorrangig testet es die Zeit, die zur Ausführung verschiedener GPI-Funktionen benötigt wird. Wenn man an hardwarenäheren Tests interessiert ist (Datentransferraten in das Video-RAM usw.), greife man lieber zu *SysBench* (s.u.), obwohl auch *SysBench* Graphiktests bietet, allerdings weitaus weniger als *PMBench*. Die Ergebnisse sind jedoch nur bedingt vergleichbar, da *SysBench* die Ergebnisse der Graphiktests in Millionen gezeichneter Pixel pro Sekunde angibt, während *PMBench* als Einheit Millisekunden benutzt. Für reine GPI-Tests ist

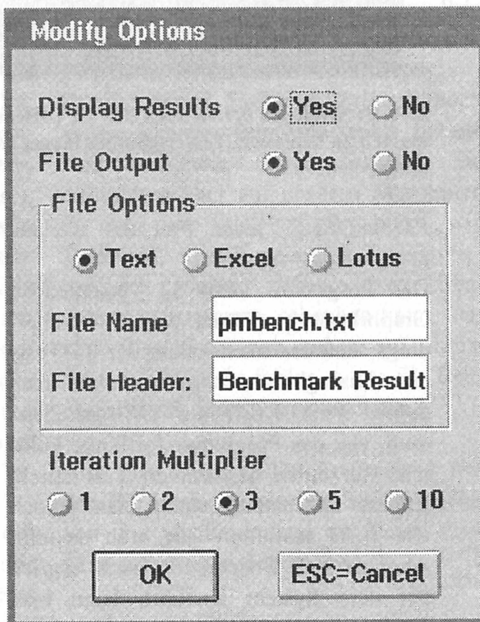


Abb. 7: Der Modify Options Dialog von PMBench

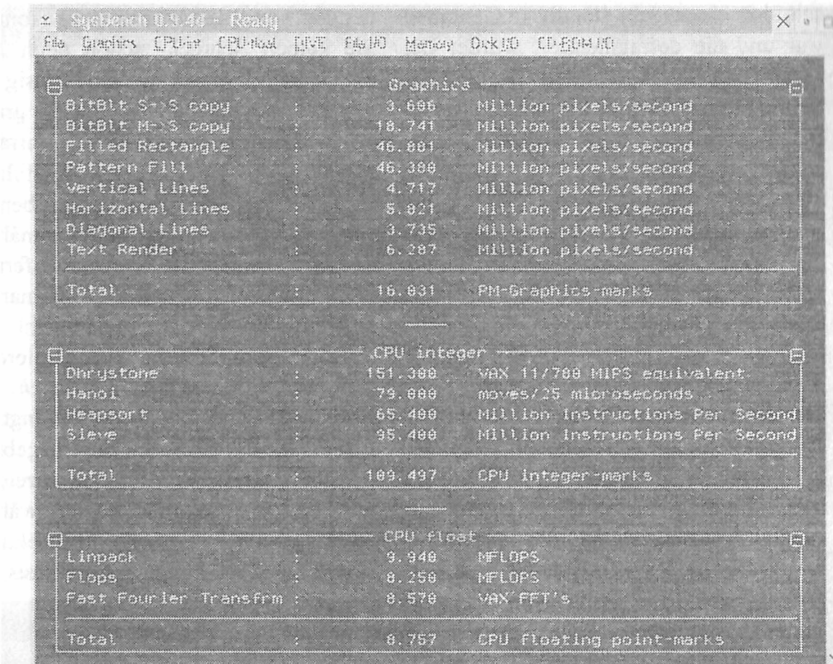


Abb. 8: SysBench 0.9.4d nach den CPU-Benchmarks. Das Programm ist übersichtlich und einfach zu bedienen. Testergebnisse lassen sich sowohl im ASCII- als auch im HTML-Format speichern.

PMBench auf jeden Fall das weitaus geeignetere Programm.

Das Programm kann 33 verschiedene Graphiktests ausführen. Negativ ist allerdings, daß das nach jedem der 33 Tests eine Dialogbox anzeigt wird, die man durch einen Klick auf *OK* schließen muß, ehe das Programm fortfährt. Führt man nur einen bestimmten Test durch, ist das akzeptabel; wenn man jedoch alle Tests ausführen läßt, stört es sehr. Aber da man Benchmarks nicht täglich auf dem System ausführt, kann man damit leben. Die Testergebnisse kann man sich auch in einer Datei sichern lassen, was man zur Durchführung aller Tests tun sollte (s. Abb. 7). Die Datei wird im Installationsverzeichnis abgelegt. Über die Dialogbox *Modify Options*, die man über den Menüpunkt *Options* erreicht, kann man außerdem angeben, wie oft ein Test wiederholt werden soll (*Iteration Multiplier*). Hier sollte man mindestens 3 angeben, um sicherere Werte zu erhalten.

Die Ergebnisse sollte man aber nicht überbewerten. Die Leistung einer Anwendung kann durchaus besser sein, als es das Benchmark vorgeben mag. Gerade im Bereich der GPI-Program-

mierung kann man viele Tricks anwenden, um z.B. den Aufbau von Graphiken zu erhöhen, womit man die tatsächliche Geschwindigkeit des GraphikEngines verschleiern kann. Und ob das Benchmarkprogramm diese Tricks nutzt, weiß man nicht. Doch für das Sammeln von Anhaltspunkten in der Bewertung eines Systems ist das *PMBench* nach wie vor zu gebrauchen. Zur Installation ist nicht viel zu sagen: Erstellen Sie auf einem Laufwerk ein Verzeichnis namens \PMBENCH, entpacken die ZIP-Datei in dieses Verzeichnis und tragen Sie es in den PATH ein (oder erzeugen Sie ein Programmobjekt mit Hilfe der WPS). Genauso gehen Sie bei der Installation des nächsten Programms vor.

Sysbench

Sysbench liegt mittlerweile in der Version 0.9.4d vor, ist weitaus aktueller als *PMBench* und testet so ziemlich alles, was der Rechner bietet: Datentransferraten über den Systembus in den Speicher, komplette Disk-I/O-Test, Graphiktests, CPU-Benchmarks usw. und alle Tests in unterschiedlichen Varianten (insgesamt sind es über 50). Man wird auch nicht durch Dialogboxen genervt. Die Testergebnisse werden alle in einem Fenster dargestellt, das man sich bequem betrachten kann (s. Abb. 8). Nach den Tests lassen sich die Ergebnisse in einer Datei sichern (entweder im ASCII-Format oder in HTML).

Während des Startens überprüft *SysBench* die Systemkonfiguration und zeigt sie in einem Notizbuch an. Sichert man die Ergebnisse eines Tests, werden diese Informationen ebenfalls in die Datei übernommen. So kann man vor allem bei vielen dieser Dateien sofort sehen, zu welcher Maschine welcher Test gehört.

Haben Sie Geduld, wenn Sie alle Tests auf Ihrem System durchführen lassen. Auf unserem 6x86 P133 brauchte *SysBench* gut 45 Minuten, um alle Überprüfungen abzuschließen. Und: Arbeiten Sie mit Ihrem System nicht, wenn das Benchmarkprogramm läuft. Ansonsten erhielte man keine aussagekrätigen Ergebnisse.

Wir sind mit *SysBench* recht zufrieden, wenngleich wir es eher selten verwenden. Gerade was die Ermittlung der I/O-Werte betrifft, so ist das Programm, wie andere Benchmarkanwendungen auch, nicht unbedingt der Verkünder der Wahrheit. Die Testsituationen des Programms stellen eben Idealsituationen dar, die in der Praxis so nie auftreten.

Es gibt neben *SysBench* übrigens noch andere kleinere Benchmarktools für die Kommandozeile (etwa *DiskIO*). Sie ermittelten aber auf unseren Maschinen in etwa die gleichen Ergebnise wie *SysBench*, so daß man sich das Hantieren mit vielen verschiedenen Benchmarkprogrammen schenken kann. Das vielseitige SysBench sollte auf Ihrem System also nicht fehlen, wenn Sie Benchmarktests durchführen möchten.

Alle hier vorgestellten Programme finden Sie in den bekannten Softwareachiven und als Begleitung zu dieser Ausgabe auf unserer Homepage. ❑

Schlanker ist schöner

OS/2 ist mit den Ansprüchen an die Systemausstattung eigentlich noch relativ bescheiden, wenn man berücksichtigt, daß es ein ausgewachsenes 32-Bit Betriebssytem ist und es mit anderen Betriebsystemen vergleicht, etwa mit Linux oder Windows NT. Trotz allem läßt sich ein Warp 4 auf einer Partition, die kleiner als 100 MByte ist, nicht installieren (obwohl ein »schlankes« OS/2 dort draufpassen würde), und man findet nach einer Installation, sei es nun von Warp 3 oder Warp 4, Dateien auf der Festplatte vor, die man ganz sicher oder in manchen Fällen nicht braucht. Die nehmen Paltz weg und sorgen nicht gerade für Übersichtlichkeit. Man kann ein Warp 3 aber bei voller Funktionsfähigkeit auf gerade einmal 18 MByte schrumpfen lassen, was den benötigten Plattenplatz angeht, und ein Warp 4 auf etwa 40 MByte. Benutzt man die WPS nicht, kann man den Festplattenanspruch weiter herunterschrauben (natürlich auch hinsichtlich der Belegung des Arbeitsspeichers). Wir zeigen Ihnen in diesem Artikel, wie man OS/2 schlanker machen kann, welche Dateien man löschen darf, welche auf gar keinen Fall, und wie sich mit anderen Maßnahmen der Ressourcenbedarf des Systems verringern bzw. optimieren läßt.

Arbeiten an der CONFIG.SYS

Wenn man nach einer OS/2-Installation zum ersten Mal in die CONFIG.SYS hineinsieht, bekommt man das kalte Grausen: Alles ist durcheinander, die Einträge befinden sich in keiner logischen Reihenfolge. Das hat zwei Nachteile:

1. Der Bootvorgang verzögert sich, weil OS/2 die CONFIG.SYS nicht sequentiell abarbeitet und folglich bei einer Startdatei, in der alles kreuz und quer verteilt ist, zuerst nach den jeweils benötigten Einträgen suchen muß.
2. Möchte man Veränderungen an der CONFIG.SYS vornehmen, findet man bestimmte Einträge nur sehr schwer.

Diese Unordnung steigt mit der Zeit immer weiter an, weil Programme bei der Installation ihre Angaben in verschiedene Bereiche der CONFIG.SYS schreiben. Besser ist es also, die wichtige Datei zunächst einmal in eine ordentliche Form zu bringen.

Obwohl das eigentlich einleuchtet, tun es viele OS/2-Anwender nicht. Bevor wir also Veränderungen an der Systemkonfiguration vornehmen, beginnen wir zunächst damit, die CONFIG.SYS zu ordnen. Gehen Sie dazu wie folgt vor:

❶ Erstellen Sie ein Backup der Datei. Öffnen Sie dann die CONFIG.SYS mit einem Texteditor, am besten dem Systemeditor, und sortieren Sie die einzelnen Eintragungen nach folgendem Schema (zwischen jeder Gruppe eine Leerzeile einfügen):

❷ Suchen Sie alle BASEDEV-Anweisungen, und stellen Sie sie an den Anfang der Datei. Sortieren Sie die einzelnen BASEDEV-Anweisungen

nach den Dateierweiterungen der Treiber in dieser Reihenfolge: SYS, ADD, FLT, DMD.

❸ Suchen Sie alle Eintragungen, die mit der Anweisung IFS beginnen, und stellen Sie sie an den Anfang der Datei.

❹ Plazieren Sie alle DEVICE-Anweisungen unter die BASEDEV-Anweisungen. Sortieren Sie die einzelnen DEVICE-Anweisungen wie folgt: Zuerst alle Treiber, die sich im Verzeichnis \OS2\BOOT befinden; anschließend die Treiber aus dem Verzeichnis \MMPM, dann alle Treiber, die sich im Verzeichnis \OS\DOS befinden. Für Warp 3 Connect oder Warp 4 fügen Sie anschließend alle Treiber aus den Verzeichnissen \IBMCOM und \MPTN hinzu.

❺ Fügen Sie anschließend unter die Treiber die LIBPATH-Anweisung, gefolgt von den SET-Kommandos für die Suchpfade PATH, DPATH, HELP und BOOKSHELF (zwischen jede Anweisung zur besseren Übersicht eine Leerzeile).

❻ Setzen Sie unter die letzte Anweisung aus *Punkt 5* diese SET-Kommandos (in der hier wiedergegebenen Reihenfolge):

SET USER_INI = ...
SET SYSTEM_INI = ...
SET RUNWORKPLACE = ...
SET OS2_SHELL = ...

Anweisungen wie:

SET RESTARTOBJECTS = ... bzw.
SET AUTOSTART = ...

und ähnliche SET-Einträge fügen Sie dem eben genannten Anweisungsblock hinzu.

❼ Suchen Sie nun sämtliche SET Befehle und reihen Sie diese unter den vorherigen Anweisungsblock. Die Reihenfolge ist dabei beliebig. Besonders wichtige Umgebungsvariablen, die mit SET definiert werden, sollten Sie allerdings in einer Gruppe zusammenfassen und durch eine Leerzeile von allen anderen trennen, etwa:

SET ETC = ...
SET NLSPATH = ... usw.

Separieren Sie ferner die Anweisungen:

SET VIDEO_DEVICES = ... sowie
SET VIO_SVGA= ...

von dem Block mit den übrigen SET-Kommandos. Ob diese vier Anweisungen vor oder hinter den anderen stehen, beeinflußt die Zeit, die OS/2 zum Booten braucht, nicht. Allerdings werden hier Informationen für den im System installierten Graphikadapter eingetragen, so daß es praktisch ist, wenn man diese Anweisungen mit einem Blick findet.

❽ Lassen Sie nun alle weiteren Kommandos wie FILES, BUFFERS, PROTECTONLY usw. folgen. Trennen Sie jedoch alle für die DOS-Unterstützung relevanten Befehle von

den anderen. Das sind: FILES, BUFFERS, SHELL, FCBS, RMSIZE und DOS.

❾ Was übrigbleibt sind die RUN- und CALL-Befehle. Setzen Sie diese an das Dateiende.

Wenn Sie gegen ein wenig mehr Mühe nichts haben, können Sie noch jeden einzelnen Abschnitt kommentieren, indem Sie eine Beschreibung hinzufügen. Diese leiten Sie mit REM ein, z.B.:

REM -------- Device Driver --------

Sie können zum Sortieren auch CFGSORT verwenden, das wir ihnen im Kapitel *Software: Werkzeuge für das System* vorstellten. CFGSRT sortiert die Einträge in der CONFIG.SYS etwas anders als wir es angegeben haben. Entsprechende Änderungen, sofern sie Ihrer Meinung nach nötig sind, können Sie nach der oben wiedergegebenen Anleitung manuell vornehmen.

Nicht benötigte Einträge und Treiber

Nachdem die CONFIG.SYS sortiert ist, findet man sich in der Datei besser zurecht und kann Veränderungen vornehmen. Wir beginnen dabei mit dem Entfernen nicht benötigter Treiber.

OS/2 installiert standardmäßig auch auf **ISA-Bus**-Systemen den Floppy-ADD-Treiber für die **Micro Channel Architektur**. Den braucht man sicher nicht, da kaum jemand einen Computer mit Micro Channel Architektur verwenden wird. Enterfernen Sie daher folgenden Eintrag aus der CONFIG.SYS:

BASEDEV=IBM2FLPY.ADD

Löschen Sie in diesem Zusammenhang auch gleich folgende Dateien aus dem \OS2\BOOT-Verzeichnis:

CLOCK02.SYS
PRINT02.SYS
SCREEN02.SYS

außerdem im gleichen Verzeichnis die ADD-Treiber:

IBM2FLPY.ADD
IBM2IDE.ADD
IBM2ADSK.ADD
IBM2SCSI.ADD

Überprüfen Sie anschließend, welche ADD- und SYS-Treiber mit BASEDEV noch in Ihrer CONFIG.SYS eingetragen sind. Alle Treiber, die Sie dort finden, notieren Sie und können alle weiteren im Verzeichnis \OS\BOOT entfernen. Löschen Sie jedoch nicht:

IBMKBD.SYS
KBDBASE.SYS
PNP.SYS

Möchten Sie von Diskettenimages mit XDFCOPY keine Installationsdisketten mehr erstellen, entfernen Sie den Eintrag:

BASEDEV=XDFLOPPY.FLT

aus der CONFIG.SYS und die Datei:

XDFLOPPY.FLT

aus dem Verzeichnis \OS2\BOOT. Sehen Sie dann in der CONFIG.SYS nach, welche FLT-Treiber mit BASEDEV tatsächlich geladen werden und löschen Sie alle anderen aus dem Verzeichnis \OS2\BOOT.
Verschieben Sie die Dateien am besten zunächst in ein Temporärverzeichnis, bevor Sie sie löschen. Sollte das System nicht starten, was uns bei der beschriebenen Vorgehensweise allerdings nicht passiert ist, können Sie das System von Diskette starten und die doch noch benötigten Dateien wieder in das ursprüngliche Verzeichnis zurückkopieren.

Überflüssige Suchpfade
Viele Programme tragen das Verzeichnis, in dem sie sich befinden, während der Installation in eine oder mehrere der PATH-Anweisungen ein. Je länger der PATH ist, um so länger kann das System brauchen, wenn es nach Dateien sucht. Daher sollte man Pfade von Programmen, die man bereits deinstalliert hat, aus den Anweisungen:

LIBPATH = ...
SET PATH = ...
SET DPATH = ...
SET HELP = ...
SET BOOSHELF = ...

entfernen. Das kann manchmal sogar dringend notwendig sein, da eine PATH-Anweisung maximal 1024 Zeichen lang sein kann. Achten Sie also auch darauf, Programme in Verzeichnissen mit möglichst kurzen Namen zu installieren, und blähen Sie die Suchpfade druch neue Verzeichnisangaben nicht unnötig auf. Plazieren Sie kleine Programme wie Systemutilities und andere Tolls daher in Pfaden wie \OS2\APPS oder \OS2.

Speicherbedarf reduzieren
Sie können den Speicherbedarf des Systems beträchtlich reduzieren, indem Sie einige Einstellungen in der CONFIG.SYS anpassen.
Verwenden Sie auf Ihren Festplatten nur HPFS, entfernen Sie den Eitnrag:

DISKCACHE = ...

aus der CONFIG.SYS, da er nur zur Bereitstellung eines Diskcaches für FAT-Laufwerke nützlich ist und bei einem reinen HPFS-Festplattensystem nur Arbeitsspeicher wegnimmt.
An dem Parameter /CACHE der der IFS-Anweisungen nehmen Sie ebenfalls Änderungen vor, indem Sie die Cachegröße für das Dateisystem reduzieren. Wir arbeiten mit HPFS-Dateisystemcaches von höchstens 1,2 MByte Größe. Man sollte nicht dem Irrtum verfallen, die Leistung des Datentransfers innerhalb des Dateisystems steige mit der Größe des Caches proportional. Wie haben ab einer Cachegröße von 1,2 MByte keine nennenswerten Leistungssteigerungen mehr feststellen können. Das ist mit ein Grund, weshalb wir einen kleinen Cache empfehlen. Cachespeicher ist für das System reserviert und steht daher anderen Programmen nicht mehr zur Verfügung. Die Gesamtleistung des Systems steigt bei einem kleineren Cache deutlich an. Ferner sollten Sie die Eintragung:

IFS = CDFS.IFS.IFS

mit dem Parameter /C: 0 versehen. Damit schalten Sie den Cache des CD-ROM-Laufwerkes aus. Wir sind der Meinung, daß man auf einen CD-ROM-Cache durchaus verzichten kann, da man von einer CD-ROM kaum innerhalb kurzer Zeit immer wieder dieselben Dateien liest. Das System reserviert für den CD-ROM-Cache Speicher in 64 KByte großen Blöcken, standardmäßig 128 KByte. Der frei gewordene Arbeitsspeicher läßt sich für neue Treiber, die man in das System einbinden muß, oder für Anwendungen besser verwenden.

Swapping justieren

Um die Arbeiten an der CONFIG.SYS vorerst abzuschließen, optimieren Sie noch das Verhalten des Systems, wenn es Pages aus dem Arbeitsspeicher auszulagern beginnt.

Wie wir im Artikel OS/2-Speicherverwaltung noch klären werden, lagert OS/2 nicht benutzte Pages in eine Datei namens SWAPPER.DAT auf der Festplatte aus. Die Größe dieser Datei beim Systemstart kann man in der CONFIG.SYS mit Hilfe der Anweisung:

SWAPPATH = *<pfad>* *<minfrei>* *<anfang>*

im Parameter *<anfang>* festlegen. Gehen Sie dazu wie folgt vor:

❶ Arbeiten Sie mit Ihrem System wie gewohnt, und überprüfen Sie vor dem Systemabschluß die Größe der Auslagerungsdatei.

❷ Addieren Sie 2 MByte zur gemessenen Größe hinzu und setzen Sie diesen Wert (in KByte) in den Parameter *<anfang>* der SWAPPATH-Anweisung.

Sichern Sie die CONFIG.SYS und führen Sie einen Systemneustart durch, wird die SWAPPER.DAT mit der in *<anfang>* angegebenen Größe erstellt. Der Vorteil liegt darin, daß OS/2 einfach mit dem Auslagern beginnt, ohne die SWAPPER.DAT währenddessen verwalten zu müssen. Übersteigt nämlich die Gesamtgröße der auszulagernden Seiten die Größe der SWAPPER.DAT, vergrößert OS/2 die Datei um 1 MByte und organisiert die darin enthaltenen Pages neu. Den Verwaltungsaufwand, der dabei entsteht, kann man sich sparen, indem man der SWAPPER.DAT von vornherein eine den Systemansprüchen angemessene Größe gibt.

Löschen unnötiger Dateien

Während wir uns zuvor mit einer Optimierung der CONFIG.SYS und dem Arbeitsspeicher des Systems beschäftigten, und nur unnötige Treiberdateien beseitigten, werden wir nun die Festplatte etwas aufräumen. Wir gehen dabei davon aus, daß Sie ein reines Arbeitssystem verwenden, das auf unnötige Zierde verzichtet.

Selektives Löschen

Seit der Version 3.0 bietet OS/2 das Programm Selektives Löschen, mit dem Sie schon einmal grob die Teile des Systems deinstallieren können, die Sie nicht brauchen. Sie finden das Programm über die WPS unter Warp 3 im Ordner *Systemkonfiguration* bzw. unter Warp 4 im

Ordner *Installieren/Deinstallieren* im Ordner *Systemkonfiguration*. Über die Kommandozeile rufen Sie das Programm mit *unistal* auf.

Nach dem Programmstart können Sie Betriebssystemkomponenten zum Löschen auswählen, sofern Sie auf Ihrem System noch vorhanden sind. Da das Programm einfach und selbsterklärend ist, müssen wir nicht näher darauf eingehen. Getrost löschen können Sie:

- alle Spiele
- Bitmaps
- Diagnoseprogramme (sofern nicht wirklich benötigt)
- das Lernprogramm
- WinOS/2-Zubehör, Klangdateien, Bildschirmschoner
- Systemdienstprogramme wie *Objektmodule verbinden*, *Festplatte sichern*, *Sortieren*.

Daneben können Sie im Verzeichnis \OS2 folgende Dateien löschen, sofern Sie dort nach dem selektiven Löschen noch vorhanden sein sollten:

8514.RC
8514M.RC
BACKUP.EXE
cchmain.exe
CGA.RC
COMETRUN.EXE
COMP.COM
CPFIND.CMD
DISKCOMP.COM
DLFCLASS.CMD
dmipm.exe
dmisl.exe
DOCKMGR.EXE
dtrace.exe
EGA.RC
erlogger.exe
fdisk.rg
FIND.EXE
ibmpcl5.drv
ibmpcl5.hlp
instprt.exe
LD2FIX.EXE
LINK.EXE
LINK386.EXE
LOG.SYS
logofred.ico
logofylw.ico
logonred.ico
logonylw.ico
LOTUSFIX
LOTUSFIX.CMD
OS2_13.RC
OS2_20.RC
os32p5.exe
PATCHWP.CMD
pcl5_fcf.dll
PLASMA.RC
PSERROR.MSG
RC.EXE
RCPP.ERR
RCPP.EXE
README.1ST
RECOVER.COM
remoterr.exe
REPLACE.EXE
RESTORE.EXE
sguide.exe
SMINST.DAT
SRD2FIX.CMD
STRACE.EXE
SYSLOG.EXE
SYSLOGPM.EXE
trace.exe
tracefmt.exe
TRACEFMT.ICO
traceget.exe
TRADEMK.ICO
TUTINTRO.FLC
TUTORIAL.EXE

UNDELETE.COM
VGA.RC
VGAM.RC
WARPUSE.ICO
WELCOME.EXE
WIN_30.RC
XGA.RC

Aus dem Verzeichnis \OS2\DLL können Sie, sofern vorhanden, folgende Dateien löschen:

CIDLOG.DLL
COMETDLL.DLL
dmiapi.dll
dmiquery.dll
dock0.cfg
dock1.cfg
dock10.cfg
dock11.cfg
dock12.cfg
dock13.cfg
dock14.cfg
dock15.cfg
dock2.cfg
dock3.cfg
dock4.cfg
dock5.cfg
dock6.cfg
dock7.cfg
dock8.cfg
dock9.cfg
errlog.dll
INFRARED.HLP
INFRARED.PDR
MINXMRI.DLL
MINXOBJ.DLL
syslogpm.dll
TUTBAMRI.DLL
TUTHEMRI.DLL
TUTHIMRI.DLL
TUTMM.DLL
TUTNEMRI.DLL
TUTSPMRI.DLL
TUTVNMRI.DLL
WELCMMRI.DLL

Aus dem Verzeichnis \OS\BOOK können die Dateien:

OVERVIEW.INF
pmdf.inf
traceref.inf
TRADEMBK.INF

gelöscht werden, und folgende Verzeichnisse können Sie ganz vom Installationslaufwerk entfernen:

\CID
\DMISL
\IBMVESA
\HELP\GLOSS
\HELP\TUTORIAL
\OS2\BITMAPS
\OS2\ART (nach der Registierung)

wenn die Systemdiagnoseprogramme nicht gebraucht werden, außerdem:

\OS2\PDPSI
\OS2\SYSTEM\RAS
\OS2\SYSTEM\TRACE

Arbeiten Sie weder mit *UltiMail/2 Light* noch mit DOS- bzw. Windows 3.x TCP/IP-Anwendungen arbeiten, löschen Sie die beiden Verzeichnisse:

\TCPIP\DOS und
\TCPIP\UMAIL sowie generell:

\TCPIP\ARCHIVE
\TCPIP\SAMPLES

und in den Verzeichnissen \TCPIP\BIN und TCTIP\DLL alle Dateien mit dem Präfix adv*, es sei denn, Sie benutzen den *Advantis Dialer*, um sich über IBM ins Internet zu wählen. Um diese Dateien zu löschen, müssen Sie zunächst

die *UltiMail/2 Light* und die *Advantis* WPS Klassen deregistrieren oder das System ohne die WPS bzw. von Diskette starten. Ansonsten sind die DLLs gesperrt, da sie durch die WPS in Verwendung sind, und können nicht entfernt werden.

Im MMOS2-Verzeichnis löschen Sie folgende Unterverzeichnisse:

BILDER
FILME
KLANGELEMENTE

Die WinOS/2-Unterstützung abmagern
Haben Sie die WinOS/2-Unterstützung installiert, können Sie diese Dateien aus dem Verzeichnis \OS2\MDOS\WINOS2 löschen:

CALC.EXE
CALC.HLP
CALENDAR.EXE
CLENDAR.HLP
CANYON.MID
CARDFILE.EXE
CARDFILE.HLP
CHARMAP.EXE
CHARMAP.HLP
CHIMES.WAV
CHORD.WAV
CLIPBRD.EXE
CLIPBRD.HLP
CLOCK.EXE
DRWATSON.EXE
GLOSSARY.EXE
MPLAYER.EXE
MPLAYER.HLP
PACKAGER.EXE
PACKAGER.HLP
PBRUSH.DLL
PBRUSH.EXE
PBRUSH.HLP
REG.COR
REG.DAT
REGEDIT.EXE
REGEDIT.HLP
REGEDITV.HLP
SOUNDREC.EXE
SOUNDREC.HLP
WRITE.EXE
WRITE.HLP

Tip: *Löschen Sie zuerst die entsprechenden Programme und Programmgruppen im Programmanager, ehe Sie die Dateien von der Festplatte entfernen.*

Nach diesen Maßnahmen hat man ein Warp 4 inkl. Java, TCP/IP, MMPM und WPS auf etwa 130 MByte gebracht. Ein Warp 3 mit DOS-Unterstützung, WPS, MMPM und IAK mißt gerade noch 75 MByte.

WPS entfernen
Wenn Sie die WPS nicht verwenden und statt dessen mit einer alternativen Shell oder CMD.EXE als Arbeitsoberfläche arbeiten, können Sie die WPS-Dateien auch noch von der Platte entfernen und sich somit um mindestens noch einmal 6 bis 12 Mbyte erleichtern (je nachdem wie groß die Archive sind, welche die WPS anlegt). Allerdings können Sie dann die WPS nicht mehr mit dem Kommando *pmshell* von der Kommandozeile aus starten. Eine alternative Shell zu verwenden, spart auf jeden Fall Arbeitsspeicher, und zwar soviel, daß man dies subjektiv merkt. Wenn Sie sich von der WPS ganz trennen möchten, löschen Sie folgende Verzeichnisse auf dem Installationslaufwerk:

\OS2\ARCHIVES
\BASISARBEITSOBERLÄCHE
\ARBEITSOBERFLÄCHE

aus dem Verzeichnis \OS2 die Dateien:

LOCK.RC
PATCHWP.CMD
PMCHKDSK.EXE
PMFORMAT.EXE
PMFORMAT.rg
WPCLS.IMP
WPCONST.CMD
WPDSACT.EXE
WPDSINIT.EXE
WPFIND.CMD
WPREXX.IMP
WPSINI.WPS
WPSINST.CMD
WPSYSOBJ.CMD

aus dem Verzeichnis \OS2\DLL die Dateien:

WPPRTMRI.DLL
WPPRINT.DLL
WPINTMRI.DLL
WPINSTAL.DLL
WPINET.DLL
WPDSRVP.DLL
WPDSERV.DLL
WPCONMRI.DLL
WPCONFIG.DLL
WPCOMET.DLL

aus dem Verzeichnis \OS2\ETC die Dateien:

WPDSERV.IR
WPSH.IR

und aus dem \OS2\HELP-Verzeichnis die Dateien:

WPHELP.HLP
WPINET.HLP
WPINSDEV.HLP
WPINSOBJ.HLP
WPINSUSR.HLP
WPMSG.HLP
WPPRINT.HLP

Einige Bemerkungen zum Schluß

Mit der hier beschriebenen Schlankheitskur waren wir mit unseren Systemen bedeutend zufriedener, sowohl was den Verbauch von RAM als auch Festspeicher anging. Unser Warp 4 Testsystem beanspruchte nur noch 112 MByte auf der Festplatte, das Warp 3 System 64 MByte, und beide Systeme liefen spürbar schneller. Man kann das System natürlich noch weiter seiner Dateien berauben. Eine einheitliche Beschreibung dürfte aber unmöglich sein, da die Systemkonfigurationen einfach zu verschieden sind. Absolute »Slim«-OS/2-Konfigurationen stellen wir Ihnen in der nächsten Ausgabe vor, wenn es um das Erstellen von Wartungspartitionen geht. Stoff zum Experimentieren haben Sie also, wenn Ihnen das Betriebssystem nicht schlank genug sein sollte. Sichern Sie aber, wie bereits geraten, die Dateien, die sie entfernen möchten, vor dem Löschvorgang, damit Sie eine Option zur Wiederherstellung haben. ❑

Tips & Tricks

In diesem Abschnitt des Know how Teils der OS/2 Only! finden Sie stets Tips & Tricks zu OS/2. Wenn Sie selber einen Trick auf Lager haben, schreiben Sie ihn uns einfach: Per eMail, Post oder auch per Fax. Veröffentlichte Tips & Tricks werden auch belohnt: Je nach Umfang gewähren wir ein Honorar von DM 25 bis DM 50. Wir und sicher alle anderen freuen sich über Ihre Mitarbeit.

Kommandozeile

Eingabe langer Dateinamen (K001)

Von der WPS her ist man es gewohnt, Objekten aussagekräftige Namen zu geben, die mitunter recht lang werden können. Von der Kommandozeile aus kann man die Eingabe aber auch bei langen Dateinamen wie gewohnt verkürzen, eine Eingabe wie:

dir "OS!2 Only Ba"*

für die Suche nach der Datei *OS/2 Only! Band 1* ist möglich. Daneben ist es nicht erforderlich, den Dateinamen mit einem zweiten " einzuschließen. Als Befehl würde also genügen:

*dir "OS!2 Only Ba**

Das gilt aber nur für den letzten in einem Befehl angegebenen Dateinamen. So würde z.B. der erste Dateiname im Befehl rename vollständig in Anführungszeichen einschlossen werden müssen, der letzte aber nicht:

rename "Bericht Mo." "Bericht Di.

PM

Systemmenüs mit der Tastatur anzeigen (P001)

Ehe man die Maus bewegt, sollte man lieber die Tastatur benutzen. Besonders Neulinge vermeiden es aber, Tastaturkürzel zu verwenden.

Die wichtigsten Funktionen eines Fensters erreicht man über dessen Systemmenü. Man kann es mit der Maus anzeigen, oder schneller mit den Tasten:

SHIFT + ESC oder
ALT + SPACE

Drückt man anschließend:

ESC oder ALT

verschwindet das Systememenü wieder. Das Systemmenü eiens Kindfensters innerhalb eines Programms erreicht man mit

STRG+ALT+SPACE

und kann es ebenfalls mit ALT oder ESC wieder verlassen.

Einmal angezeigt, kann man die einzelnen Menüpunkte schnell mit ihren Buchstabenkürzeln auswählen. Besonders praktisch ist dies bei Systemmenüs, die verschiedene Funktionen eines Programms zur auswahl bieten (wie bei Impos oder den Ordnern der WPS).

Hardware

System läßt sich nicht mehr von IDE-Festplatte starten (H001)

Falls das System nach dem Anzeigen der Systemkonfiguration nicht mehr von der Festplatte startet und auch der Bootmanager nicht mehr angezeigt wird, vermutet man einen Defekt der Festplatte oder des IDE-Controllers. Bevor man jedoch panisch die Hardware überprüft, sollte man zunächst von Diskette starten und versuchen, auf die Festplatte zuzugreifen. Verblüffenderweise ist das manchmal nämlich noch möglich.

Erhält man einen Zugriff, kann man das System wieder zum Starten bringen, indem man *alle* Einträge in der Partitionstabelle löscht und anschließend wieder neu einfügt. Dazu sollte man natürlich ein Backup parat haben. Hat man keines, fertigt man mit Hilfe der Notfalldisketten ein Backup aller Partitionen an.

Anschließend ruft man FDISK auf und löscht alle Partionen, um die Platte anschließend neu zu partitionieren. Achten Sie bitte darauf, die Partition, welche das Betriebssystem aufnehmen soll, mit FDISK als *Startbare Partition* zu markieren und in das Startmenü des Bootmanagers einzutragen. Alternativ dazu können Sie aber auch eine Installationspartition festlegen und das Betriebssystem neu installieren. Nach der Partitionierung ist ein Neustart erforderlich.

Man bootet wieder von Diskette, formatiert die Partitionen und überträgt die gesicherten Daten wieder auf die Harddisk; oder, hat man sich für eine Neuinstallation des Betriebssystems entschlossen, man startet mit den Installationsdisketten und installiert zunächst OS/2, bevor man die Daten zurückspielt.

Nach dieser wenn auch umständlichen Prozedur wird bei einem Neustart der Bootmanager wieder angezeigt und das System startet wie gwohnt.

Lediglich den Bootmanager zu entfernen und neu einzurichten, führt allerdings zu keinem Erfolg; ebensowenig wie das Löschen aller startbaren Partitionen (um wenigstens die Datenpartitionen unberührt zu lassen). Wohl oder übel muß man die gesamte Partitionstabelle neu aufbauen, um das System zum Starten zu bringen.

Verloren sind die Daten in einem solchen Fall vorerst wie gesagt noch nicht. Sie können die Platte für Backupzwecke sogar als Slave konfigurieren und an einen anderen Controller anschließen. Nur ein Start ist mit der Platte nicht mehr möglich.

Software

Mehrere Konfigurationen bzw. Benutzer mit Netscape 2.02 (S001)

Häufig steht man vor dem Problem, mehrere Mail-Konfigurationen zu benötigen; sei es, weil man mehrere eMail-Adressen hat oder private von geschäftlicher Mail trennen und andere persönliche Angaben und Signaturdateien verwenden möchte.

Der Weg dorthin führt über das Anlegen mehrerer *netscape.ini*-Dateien und das Erstellen verschieden konfigurierter Icons. Der Schüssel zu dieser Funktion ist der *-i* -Parameter von *Netscape*, mit dem man dem Programm verschiedene INI-Dateien mit unterschiedlichen Pfaden beim Start übergeben kann.

Zuerst sichert man die bestehende *netscape.ini* (z.B. als *netscape.bak*). Dann kopiert man die *netscape.ini*-Datei so oft, wie Konfigurationen benötigt werden, und benennt die Kopien entsprechend um, z.B. in *netscape.privat* und *netscape.firma*. Beim Start von *Netscape* wird automatisch die im *Netscape*-Pfad befindliche *netscape.ini* benutzt.

Für mehrere Konfigurationen wird nun das *Netscape*-Icon entsprechend oft kopiert und gleich wie gewünscht benannt (in unserem Beispiel *Mail privat* und *Mail Firma*). Anschließend werden die Eigenschaften eines jeden neuen Icons aufgerufen. Hier erfolgt nun der entscheidende Eintrag in der Parameter-Zeile:

-inetscape.privat -mail und
-inetscape.firma -mail

! ***Achten Sie darauf, daß nach dem Parameter* -i *kein Leerzeichen folgt.***

Der Parameter *-mail* sorgt für einen sofortigen Start des Mail-Fensters. Mit Pfadangaben könnte der Eintrag dann so aussehen:

-ic:\netscape\konfigs\netscape.privat.

Zum Schluß wird *Netscape* über die verschiedenen Icons aufgerufen und entsprechend konfiguriert. Nicht vergessen: Legen Sie zur Vorsicht nach Abschluß der Konfigurationsarbeiten Sicherheitskopien der INI-Dateien an. Diese Funktion läßt sich natürlich auch einsetzen, um *Netscape*-Versionen mit und ohne JAVA bzw. JIT-Compiler zu starten; oder um in Netzwerken eine Multi-User-Konfiguration von *Netscape* zu ermöglichen. So muß in diesem Fall die *Netscape*-INI im Homelaufwerk des Benutzers abgelegt sein.

(Heiko Korsawe, Groitzsch)

OS/2-Speichermanagement

In nahezu jedem Buch zur OS/2-Programmierung findet man den ein oder anderen mehr oder weniger umfangreichen Abschnitt, der das Speichermanagement von OS/2 darstellt. Ausführliche Beschreibungen bleiben aber aus. Da ein Wissen um die Art und Weise, wie OS/2 mit dem Systemspeicher umgeht, nicht nur wichtig für die Programmierarbeit ist sondern auch für jeden Benutzer, möchten wir mit diesem Artikel versuchen, eine Lücke in der deutschsprachigen OS/2-Literatur zu schließen. Nicht zuletzt wollen wir noch klären, wie OS/2 auf der bereits beschriebenen Hardwareplattform, dem 80386, aufsetzt.

Nachteile des segmentierten Speichers

Im Artikel *Die CPU* lernten wir den 80386 kennen und das *segmentierte Speichermodell*, das er implementiert. Segmente bieten einige Vorteile, etwa daß Programme mit Selektoren arbeiten, um ihre Speicherbereiche zu adressieren, wohingegen die linearen Adressen im Deskriptor stehen, auf die nur das Betriebssystem Zugriff hat. Damit sind die Segmente aller Prozesse im System voneinander getrennt. Weitere Vorteile sind das Konzept des virtuellen Speichers durch das Ein- und Auslagern von Segmenten auf die Festplatte und die Limit-Überprüfung des Segmentes, wodurch ein Programm nicht auf Speicherstellen jenseits der Segmentgrenze zugreifen kann. OS/2 1.x war für den 80286 programmiert und machte vom segmentierten Speichermodell dieses Prozessors Gebrauch. Die Segmentierung bringt aber auch schwerwiegende Nachteile sich:

- Immer wenn auf ein Segment zu griffen wird, muß der Segm Selektor zur Adreßübersetzung in entsprechende Segmentregister g den werden. Der Zeitaufwand, Selektoren in die Segmentregister laden, wird durch die Schutzüber fungen, die der Prozessor durchfü und durch den Zugriff auf Deskriptor nach Laden des Segm tregisters noch erhöht, womit Ausführungszeit beim Wechsel Segmenten beim 80386 etwa 18 Takte beträgt.
- Der physikalische Speicher fragm tiert beim Ein- und Auslagern Segmenten. Da Segmente un schiedliche Größen haben, entste mit der Zeit unbenutzte Bereiche z schen den einzelnen Segmenten Speicher (siehe Abb. 1). Daher r das Betriebssystem den physik schen Speicher regelmäßig neu nen (defragmentieren), was ei erhöhten Verwaltungsaufwand (O head) bedingt.
- Segmente können unangenehm g werden, so daß Schwierigkeiten b Ein- und Auslagern entstehen.
- Es muß mit 48-Bit Zeigern gearbe werden, die eine aufwendige Beha lung durch den Programmierer er dern.
- Das Programm muß entspreche Verwaltungsstrukturen anlegen, mit Segmenten effizient umgehen

können. Sowohl die Programmgröße als auch die Leistung der Anwendung wird dadurch negativ beieinflußt.

OS/2 und das Flat Memory Model

Es wäre daher wünschenswert, die Segmentierung zu deaktivieren und mit einem linearen Adreßraum zu arbeiten, der bei 0 beginnt und den gesamten adressierbaren Speicherbereich, also 4 GByte umfaßt. Dann könnte man direkt mit dem 32-Bit Offset den gesamten Adreßraum ansprechen, und ein Arbeiten mit den 48-Bit Zeigern und dem zeitintensiven Übersetzungsmechanismus entfiele.

OS/2 implemeniert ab der Version 2.0 ein solches lineares Speichermodell, das man auch **Flat Memory Model** nennt. Allerdings bietet der Prozessor keine Möglichkeit, die Segmentierung abzuschalten (was auch unsinnig wäre, da die Schutzkonzepte auf dem Segmentkonzept basieren). Man kann jedoch einen flachen Adreßraum auf dem 80386 simulieren. OS/2 macht das so:

- Während der Initialisierung des Systems erzeugt OS/2 ein Code- und ein Datensegment und setzt dabei die Basisadresse jedes Segmentselektors auf 0.
- Der Segmentselektor wird in jedes der 6 Segmentregister geladen, wodurch alle Register auf den gleichen Speicherbereich verweisen.
- Die Segmentselektoren werden innerhalb der GDT bereitgestellt; damit kann nur das Betriebssystem auf sie zugreifen.

So entsteht ein einziger Adreßraum von 4 GByte Größe, da nur noch der 32-Bit-Offset der 48-Bit virtuellen Adresse benötigt wird bzw. verwendet werden kann, um einen Adreßraum von 0 beginnend bis 4 GByte anzusprechen. Auf diese Weise kann mit »handlichen« 32-Bit-Zeigern gearbeitet werden, wodurch sich das Speichermanagement in Anwendungen erheblich vereinfacht, und die durch die Segmentierung entstandenen Nachteile sind beseitigt. Da nur die Offsetkomponente der virtuellen Adresse zur Adressierung verwendet wird, müssen die Seg-

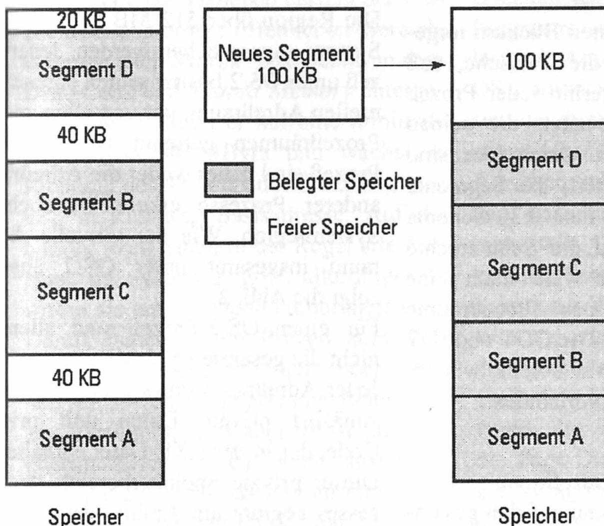


Abb.1: Speicherfragmentierung beim Segmentmodell

mentregister bei Speicherzugriffen nicht neu geladen werden, und so wird auch der Performancegewinn klar, den OS/2-Anwendungen gegenüber 16-Bit-Programmen aufweisen. Dennoch werden die durch die Segmentierung eingeführten Privilegebenen weiter beibehalten. Die Selektoren sind demnach keine Dummy-Zeiger, lediglich die Basisadresse aller Segmente ist 0. Daß die Selektoren nicht mehr zur Adreßübersetzungs verwendet werden müssen, liegt daran, daß sie alle auf den gleichen linearen Adreßraum verweisen, ohne jedoch die Schutzmechanismen zu deaktivieren, die im Selektor resp. dem Deskriptor enthalten sind. Damit kann OS/2 trotz des flachen Adreßraumes die im Kapitel *Die CPU* beschriebenen Privilegebenen weiterhin nutzen. OS/2 benutzt den Ring 0 für den Kernel des Systems; Ring 2 für bestimmte Treiber und Serviceprogramme und Ring 3 für Anwendungen. Ring 1 wird nicht benutzt.

Was auch in den meisten Büchern totgeschwiegen wird, ist die Tatsache, daß auch unter OS/2 weiterhin jeder Prozeß seine eigene LDT besitzt, die seinen Adreßraum beschreibt. Auch hier sind Deskriptoren gespeichert, die Segmente beschreiben, die nach dem 0:32 Schema adrresiert werden und die Schutzrechte bestimmen.Wie sonst wäre auch eine Trennung der einzelnen Prozeßräume voneinander möglich? Die GDT und IDT sind natürlich, wie wir bereits wissen, nur einmal im System vorhanden.

Globaler- und Prozeßadreßraum

Das Betriebssystem kann auf den gesamten linearen Adreßraum zugreifen. Theoretisch kann auch jeder Prozeß unter OS/2 einen 4 GByte großen Speicherbereich adressieren. OS/2 limitiert den Adreßraum eines Prozesses aber auf 512 MByte und zwar aus zwei Gründen:

1. Um die Kompatibilität mit 16-Bit OS/2-Anwendungen sicherzustellen, die nur einen maximalen Adreßraum von 512 MByte ansprechen können. Diese Anwendungen laufen daher ohne Veränderungen auf einem 32-Bit OS/2-System. Daher werden die ersten 512 MByte des virtuellen Adreßraums auch als Kompatibilitätsregion bezeichnet.

2. Um Anwendungen den Zugriff auf die Speicherregion jenseits der 512 MByte-Grenze zu verweigern, die man daher auch Systemregion nennt und nur dir durch 32-Bit-Adressen angesprochen werden kann.

Die Region über 512 MB kann nur vom System angesprochen werden. Jeder Prozeß unter OS/2 besitzt seinen eigenen virtuellen Adreßraum, der von allen anderen Prozeßräumen getrennt ist. Für einen Prozeß sind daher weder die Adreßräume anderer Prozesse erkennbar, noch die Systemregion. Wie der virtuelle Adreßraum insgesamt unter OS/2 aussieht, zeigt die Abb. 2.

Für einen OS/2-Prozeß sind allerdings nicht die gesamten 512 MByte verfügbar. Jeder Adreßraum eines Prozesses enthält zunächst private Daten und privaten Code, der in der EXE-Datei enthalten ist. Dieser private Speicherbereich des Prozesses beginnt am Anfang des virtuellen Adreßraumes und hat eine minimale

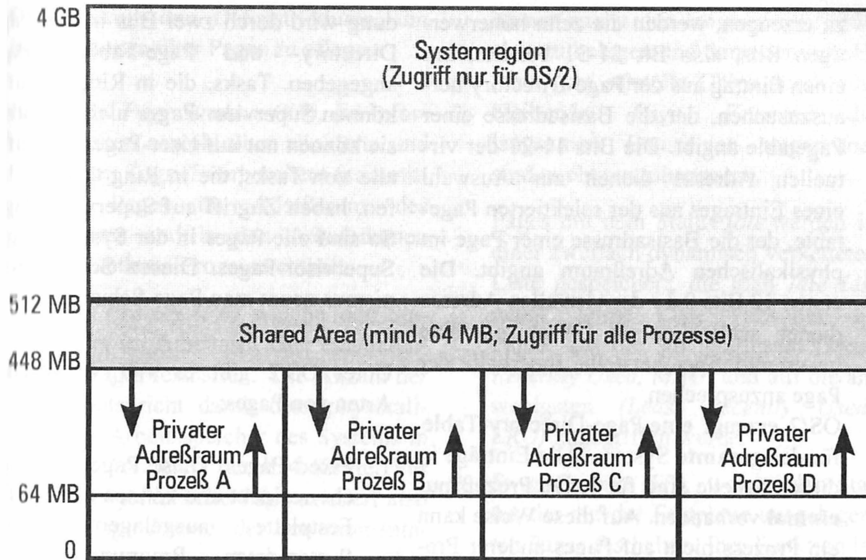


Abb. 2: Der virtuelle Adreßraum von OS/2 und der Aufbau der virtuellen Prozeßadreßräume

Größe von 64 MByte. Der private Speicherbereich wächst in Richtung der 512 MByte-Grenze. Daneben besteht der Prozeßraum aus einem öffenlichen Bereich, den man auch *Shared Area* nennt, in dem DLL-Code und *Shared Memory* untergebracht wird. Auch er hat eine minimale Größe von 64 MByte und wächst in Richtung des Anfangs des Prozeßraumes, also in Richtung Basisadresse. Auf die *Shared Area* haben in der Regel alle Prozesse im System Zugriff, allerdings nur sofern sie entsprechend authorisiert sind. Damit stehen einem Prozeß zum Programmstart maximal nur noch 448 MByte Speicher zur privaten Benutzung zur Verfügung (was aber selbst für speicherintensive Anwendungen ausreicht). Abbildung 2 stellt das Layout eines Prozeßadreßraumes unter OS/2 dar.

OS/2 und das Paging

Weil unter OS/2 das Paging aktiv ist, entsprechen die linearen Adressen, die ein Programm verwendet, nicht den physikalischen Adressen. Jeder Prozeß unter OS/2 arbeitet also mit virtuellen Adressen, die durch den *Paging-Adreßübersetzungs-Mechanismus* in physikalische Adressen umgewandelt werden. Neben den voneinander getrennten Adreßräumen stellt OS/2 mit dem Paging einen zweiten Schutzmechanismus, den Page-Schutz zur Verfügung.

Wie wir wissen, wird durch das Paging der gesamte Adreßraum in 4 KByte große Blöcke unterteilt, die sogenannten Pages. Zur Verwaltung der Pages stehen die Page-Directory und die Page-Tables zur Verügung. Um eine 32-Bit physikalische aus einer 32-Bit virtuellen Adresse

zu erzeugen, werden die zehn höherwertigen Bits, also Bit 21-31 benutzt, um einen Eintrag aus der Page-Directory herauszusuchen, der die Basisadresse einer Pagetable angibt. Die Bits 11-21 der virtuellen Adresse dienen zur Auswahl eines Eintrages aus der selektierten Pagetable, der die Basisadresse einer Page im physikalischen Adreßraum angibt. Die ersten 12 Bits 0-11 der virtuellen Adresse dienen schließlich als Offset, um eine bestimmte Speicherstelle innerhalb der Page anzusprechen.
OS/2 erzeugt **eine** Page-Directory-Table für das **gesamte** System. Alle Einträge in dieser Tabelle sind für **jeden** Prozeß nur **einmal** vorhanden. Auf diese Weise kann ein Prozess nicht auf Pages anderer Prozesse im Arbeitsspeicher zugreifen. Bei jedem Taskwechsel werden nun die prozeßeigenen Einträge in die Page-Directory-Tabelle geladen, die dann auf die entsprechenden Page-Tables verweisen. Insgesamt können 1024 Einträge in die Page-Directory-Tabelle aufgenommen werden und ebenfalls maximal 1024 Einträge faßt eine Page-Table. Damit kann eine Page-Table 1024 Pages, also 4 MByte adressieren und maximal 1024 Page-Tables sind in die Page-Directory-Tabelle eintragbar, womit der gesamte lineare Adreßraum abgedeckt wäre (wobei natürlich nicht alle Einträge benutzt werden, da der virtuelle Adreßraum eines Prozesses ja auf 512 MB beschränkt ist).
Auf der Ebene der Pages gibt es zusätzlich zu den bereits angesprochenen Privilegbenen einen weiteren Schutzmechanismus: Eine Page kann eine User- oder Supervisor-Page sein. Die Unterscheidung wird durch zwei Bits in den Page-Directory- und Page-Table-Einträgen angegeben. Tasks, die in Ring 3 laufen, können Supervisor-Pages nicht benutzen, sie können nur auf User-Pages zugreifen, alle von Tasks, die in Ring 0 und 2 laufen, haben Zugriff auf Supervisor-Pages. So sind alle Pages in der System-Region Supervisor-Pages. Diesen Schutzmechanismus nennt man Page-Schutz.

Unter OS/2 gibt es vier verschiedene Arten von Pages:

1. Fixed Pages: Diese Pages sind speicherresident und können nicht auf die Festplatte ausgelagert werden. Systemdaten, Programmcode des OS/2-Kernels, Treiber oder Cachespeicher werden in Fixed Pages im Speicher abgelegt.

2. Swappable Pages: Diese Pages können, wird auf sie längere Zeit nicht zugegriffen, vom Arbeitsspeicher auf die Festplatte ausgelagert werden, wenn physikalischer Speicher benötigt wird.

3. Discardable Pages: Diese Pages können ganz aus dem Speicher entfernt werden, ohne daß OS/2 sie auslagert. Das betrifft lediglich Code, der aus EXE- und DLL-Dateien erneut in den Speicher geladen werden kann, wenn er benötigt wird.

4. Invalid Pages: Diese Pages wurden von einem Prozeß zwar allokiert, stehen ihm aber physikalisch noch nicht zur Verfügung. Solche Pages bezeichnet man als *uncommitted.*

OS/2 erzeugt drei Datenstrukturen, um Informationen über Pages zu pflegen:

1. *Virtual Page Struktur (VP),* die Informationen sowohl über allokierte und für den Zugriff vorbereitete (committed) Pages im Adreßraum des Systems und in den Adreßräumen aller anderen Prozesse enthält.

2. *Page Frames (PF),* welche den Status der im Arbeitsspeicher existierenden Pages festhalten. Die Anzahl der PF entspricht damit dem physikalischen Arbeitsspeicher des Systems in KByte dividiert durch 4 Kbyte. Für jede physikalische Page existiert also ein *Page Frame,* der den momentanen Status einer Page im Speicher wiedergibt. Eine Page hat entweder den Status *Free,* dann steht sie einem Prozeß zum Allokieren zur Verfügung; den Status *In-use,* dann wird die Page gerade von einem Prozeß verwendet; oder den Status *Idle,* was bedeutet, daß die Page von einem Prozeß allokiert wurde, aber kein *Page-Table-Eintrag* des gerade laufenden Prozesses auf eine dieser Pages verweist (daß der Prozeß, der die Pages allokierte, also momentan nicht aktiv ist).

   Pages mit dem Status *Free* werden in einer zweifach dynamisch verketteten Liste gespeichert, die man *Free List* nennt. Diese Liste verwalted an einem Ende die *Page Frames* für langsamere Speicherbausteine und am anderen Ende die für schnelle Module. OS/2 benutzt diese Information, um bei Speicheranforderungen immer zuerst den langsameren Speicher zu belegen und dann den schnelleren (um schnelle Chips durch verbleibende *Page Frames,* die langsamen Bausteinen zugeordnet sind, nicht auszubremsen).

   Pages mit dem Status *Idle* werden in einer zweifach dynamisch verketteten Liste gespeichert, die man *Idle List* nennt. Diese Liste verwaltet die Pages, auf die am häufigsten *(Most Recently Used, MRU)* und auf die am wenigsten *(Least Recently Used, LRU)* zugegriffen wurde.

3. *Swap Frames (SF),* welche den Status der auf der Festplatte ausgelagerten Pages festhalten. Ihre Funktion ist damit eigentlich dieselbe wie die der *Page Frames,* nur das die *Swap Frames* Bereiche in der Swapdatei beschreiben, in denen Pages vom Arbeitsspeicher abgelegt werden können.

Wie OS/2 Speicher bereitstellt

Ein OS/2-Programm kann Speicher nur auf Pagebasis allokieren, also jeweils in 4 KByte großen Blöcken. Eine Page ist damit die kleinste unter OS/2 allokierbare Speichereinheit. Sowohl der virtuelle Adreßraum eines Prozesses als auch der physikalische Arbeitsspeicher ist in Pages unterteilt.

Wenn ein Prozeß in seinem virtuellen Adreßraum auf eine allokierte und für den Zugriff vorbereitete Page zugreifen möchte, stellt OS/2 nicht sofort den angeforderten Speicher zur Verfügung. Der Page-Table-Eintrag des Prozesses weist dann auf eine *VP-Struktur;* allerdings

wird die Page als im Speicher nicht vorhanden gekennzeichnet. Greift ein Prozeß dann auf eine solche Page zu, die er durch einen Eintrag in seiner Page-Table zwar definiert, die aber im Speicher nicht vorhanden ist, wird ein sogenannter *Page-Fehler* generiert, den OS/2 wie folgt bearbeitet:

1. Entweder das Betriebssystem allokiert eine physikalische Page mit dem Status *Free,* indem ein *Page Frame* aus der *Free List* zur Verfügung gestellt wird. Ist eine solche Page nicht mehr vorhanden, stellt OS/2 eine LRU-Page (mit dem Status *Idle*) aus der *Idle List* bereit. Wurden in in die *Idle Page* zuvor Daten geschrieben, schreibt OS/2 diese Page zuerst in die Swapdatei, um Datenverlust zu vermeiden.

2. Nimmt aber die Anzahl an Pages mit dem Status *Idle* und *Free* bis zu einem Grenzwert ab, da sie schon in Verwendung sind, können keine *Page Frames* mehr bereitgestellt werden. Also überführt OS/2 Pages mit dem Staus *In-use* in den *Idle*-Status, um Speicheranforderungen weiterhin behandeln zu können.

3. Um zu entscheiden, welche *In-use* Pages den *Idle Status* bekommen können, verwendet OS/2 einen sogenannten *Page ager*, der alle *In-use Pages* regelmäßig auf Zugriffe überprüft (was durch einen Check des *Access Bits* im *Page-Table Eintrag* erfolgt). Wurde seit dem letzten Test auf eine *In-use Page* nicht zugegriffen, fügt Sie OS/2 in die *Idle List* ein, und der *Page Frame* kann zur Verfügung gestellt werden.

4. Nachdem ein *Page Frame* vorhanden ist, lädt OS/2 die Informationen aus der *Virtual Page Struktur* in den *Page Frame.*

5. Die *VP Struktur* legt fest, was das Betriebssystem daraufhin zu tun hat. Ist die Page eine *Discardable Page* wird ihr Inhalt von der EXE- oder DLL-Datei neu in den Speicher geladen; ist es eine *Swappable Page* gibt es zwei Möglichkeiten: Entweder die Page befindet sich noch in der *Idle List,* dann muß ihr Inhalt nicht neu geladen werden, da sie sich noch im Speicher aufhält; oder sie wurde bereits aus dem Arbeitsspeicher entfernt. In diesem Falle lädt OS/2 den Inhalt der Page von der Swapdatei neu.

6. Nun ist die Page im Speicher verfügbar. OS/2 speichert die Basisadresse des *Page Frames* im *Page Table Eintrag* des Prozesses und markiert die Page als vorhanden.

7. Zum Abschluß wird noch der *TLB* geleert, und das Programm fährt mit seiner Ausführung fort, ohne daß irgend etwas vorgefallen wäre.

Die Swapdatei

OS/2 speichert Pages in einer Swapdatei namens SWAPPER.DAT, die in einem Verzeichnis liegt, das durch die Anweisung SWAPPATH in der CONFIG.SYS angegeben wird (wir lernten diese Anweisung bereits im Artikel *Schlanker*

Ist schöner kennen). Standardmäßig ist im SWAPPATH das Verzeichnis \OS2\SYSTEM eingetragen, und die Datei ist anfangs mindestens 512 KByte bzw. ein Vielfaches von 512 KByte groß (wobei man ihre Anfangsgröße durch den Parameter <*anfang*> in der SWAPPATH-Anweisung auch verändern kann, wie wir zeigten).

Die Größe der SWAPPER.DAT berechnet OS/2 jedesmal, wenn Speicher im System angefordert wird. Dabei addiert das Betriebssystem die Gesamtgröße des Speichers, der durch alle *Fixed Pages* belegt wird, mit dem, den alle *Swappable Pages* in Anspruch nehmen und berechnet anschließend den Betrag, um den diese Summe den tatsächlich im System installierten Speicher übersteigt. Übersteigt die Größe des benötigten Speichers die Größe des Systemspeichers, spricht man auch von einem *Overcommitment.*

Wird immer mehr Speicher benötigt, wird die Größe des Systemspeichers in Relation zum tatsächlich benötigten Speicher immer kleiner, und die SWAPPER.DAT wird entsprechend vergößert, und zwar in 512 KByte-Schritten bzw. in Vielfachen dieser Schritte. Damit muß die Datei nicht jedesmal vergrößert werden, wenn ein *Overcommitment* festgestellt wird.

Wenn nun eine Page auf die Festplatte ausgelagert werden soll, benutzt OS/2 ein Array aus *Swap Frames*, um Pages freie Stellen in der SWAPPER.DAT zuzuweisen. Solange die Datei groß genug ist, um das *Overcommitment* abzufangen, werden diese Pages einfach in die Datei geschrieben; ansonsten wird die Datei vergrößert. Werden nun etliche Pages nicht mehr benötigt, wird wieder Platz in der SWAPPER.DAT frei, und OS/2 registriert bei der Berechnung des *Overcommitments*, daß die SWAPPER.DAT zu groß ist, weshalb das System die Datei wieder verkleinert. Dazu ist ein gewisser Verwaltungsaufwand erforderlich, und die SWAPPER.DAT schrumpft nicht so schnell, wie Sie wächst. Nun wird auch der Rat einsichtig, warum die SWAPPER.DAT möglichst dem typischen *Overcommitment* des Systems angepaßt sein sollte. Eine Vergrößerung und ein Schrumpfen der Datei tritt nur dann ein, wenn die Anfangsgröße überschritten wird.

Zum Schluß

Mit Paging wird das Konzept des virtuellen Speichers realisiert: Pages können aus dem Arbeitsspeicher auf die Festplatte ausgelagert und von dort wieder eingelagert werden. Durch die einheitliche Größe geht das bedeutend schneller vonstatten als mit Segmenten. Zudem tritt keine Speicherfragmentierung auf, da bei der Benutzung des physikalischen Speichers mit der Zeit immer nur 4 KByte große »Löcher« entstehen, die durch andere Pages gefüllt werden können. Die einzelnen Pages eines Prozesses können dabei kreuz und quer im Speicher verteilt sein; es ist nicht notwendig, daß zusammenhängende Blöcke entstehen müßten. Das Programm merkt von der wirklichen Anordnung seiner Pages nichts, da es ja mit virtuellen linearen Adressen arbeitet (s. Abb. 3). Es sieht seinen Adreßraum als großen, zusammenhängenden Bereich von 512 MByte

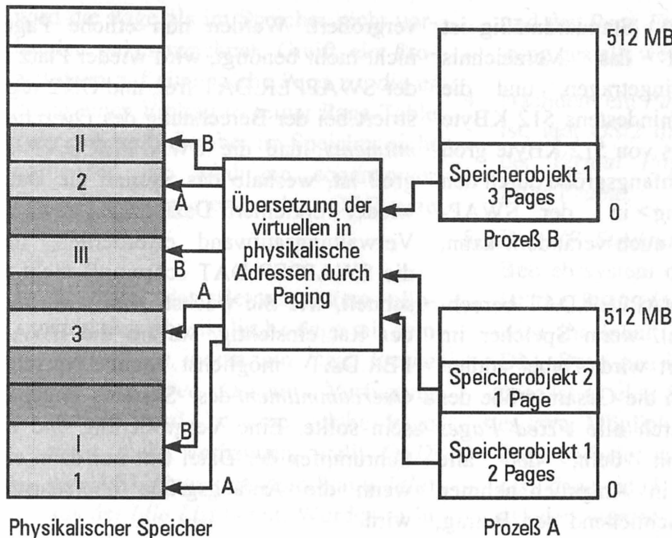


Abb. 3: Verteilung der Pages im Speicher. Die Anordnung ist völlig beliebig, und eine Fragmentierung wird vermieden.

Größe, der natürlich auch in 4 KByte große Blöcke unterteilt ist. Das ist das einzige, worauf man bei der Programmierung achten muß: Speicher wird vom System immer in 4 KByte-Blöcken zur Verfügung gestellt, da dies die Maßeinheit ist, mit der der Speicher granuliert wird. Der Entwickler allokiert Speicher als sogenannte Speicherobjekte die aus mindestens 4 KByte bzw. einem Vielfachen aus 4 KByte bestehen. Was es sonst noch zu beachten gibt, werden wir in einer der folgenden Ausgaben in einem Artikel zur *Speicherverwaltung* im Kapitel *Programmieren* zeigen.

Durch Paging ist die Größe des virtuellen Speichers nur durch die Größe der Festplatte limitiert, welche die ausgelagerten Speicherseiten aufnimmt. Damit können selbst mehrere Prozesse, die vom Speicher intensiven Gebrauch machen, auf einem System mit wenig RAM ausgeführt werden. Alle nicht benötigten Pages werden ausgelagert, die gerade benötigten eingelagert. Allerdings vergrößert sich der Verwaltungsaufwand für das Swapping, je kleiner der im System installierte physiklische Speicher ist. Das kann soweit gehen, daß das System nur noch mit dem Swappen beschäftigt ist. Ein unangenehmer Vorgang, den man als »Trashing« oder Flattern bezeichnet.

Ein System mit möglichst viel RAM auszustatten, ist also nach wie vor der beste Rat zur Aufrüstung eines Rechners. Trotz allem kann das beste Betriebssystem und die schnellste Hardware Programmschwächen nicht ausgleichen. Als Programmierer achte man man also darauf, so schlank und speicherschonend wie möglich zu entwickeln. ❑

Workshop DLLs, Teil 1

Dynamische Linkbibliotheken, kurz **DLLs**, stellen einen der wichtigsten Bereiche der OS/2-Anwendungs- und Systemprogrammierung dar und eröffnen Entwicklern vielfältige Möglichkeiten, die wir in einem dreiteiligen Workshop betrachten möchten.

Im ersten Teil dieser Reihe vermitteln wir theoretisches Basiswissen, über das man verfügen sollte, und geben Anleitungen zum Entwerfen eigener, einfacher DLLs, die Funktionen und Ressourcen enthalten; im zweiten Teil erweitern wir unser Beispiel, indem wir ein REXX-Interface hinzufügen, um einer Applikation die Möglichkeit zur Definition von REXX-Makros zu geben und zeigen, wie man Funktionsbibliotheken für REXX erzeugt; im letzten Teil entwickeln wir mit Hilfe einer DLL ein einfaches Subsystem und runden damit die Betrachtung der DLL-Programmierung ab.

Alle Programmbeispiele in diesem Artikel sind in C codiert. Wir wählten C, da es für die OS/2-Programmierung die geeignetste Sprache ist, und C-Code auch zusammen mit C++ genutzt werden kann. Daneben dürfte eine Adaptierung auf andere Sprachen einfach sein. Wir verwenden das *C-Set* von IBM mit dem *Warp 3 Toolkit*. Aus Platzgründen können wir nicht den gesamten Beispielcode abgedruckt wiedergeben. Sie finden ihn aber komplett auf unserer Homepage als ZIP-Datei zum Download.

Ein wenig Theorie

DLLs ermöglichen Programmen, Funktionen zu benutzen, die außerhalb des EXE-Moduls definiert sind und während der Laufzeit in den Speicher geladen und mit einem Prozeß verbunden werden, was man als **Dynamic Linking** (im Gegensatz zum **Static Linking**) bezeichnet. Ist die DLL einmal in den Speicher geladen worden, stehen allen Prozessen im System die exportierten Funktionen zur Verfügung, d.h.:

- oft benötigte Funktionen (oder auch Ressourcen wie Menüs oder Dialoge) müssen nur einmal in den Speicher geladen werden, wodurch der Speicherverbrauch von Programmen gleicher Codebasis reduziert werden kann,
- die Größe der EXE-Dateien sinkt,
- die Ladezeiten der EXE-Module werden verringert.

Da man mit DLLs eine Kapselung des Funktionscodes erzielt, können DLLs unabhängig vom EXE-Modul weiterentwickelt werden. Dadurch kann man die Funktionalität für unterschiedliche EXE-Dateien alleine durch Veränderung der DLLs erhöhen. Ein gutes Beispiel dafür ist der **Presentation Manager**, der zum größten Teil aus DLLs besteht, welche die Funktionalität des OS/2 Kernels zum einem GUI-System erweiterten.

Speichermanagement

Eine DLL enthält ausführbaren Programmcode, ist aber keine ausführbare Programmdatei, d.h. sie kann kein Prozeß werden, wenn sie geladen wird. Folgerichtig werden Funktionen, die in einer DLL enthalten sind, innerhalb des Pro-

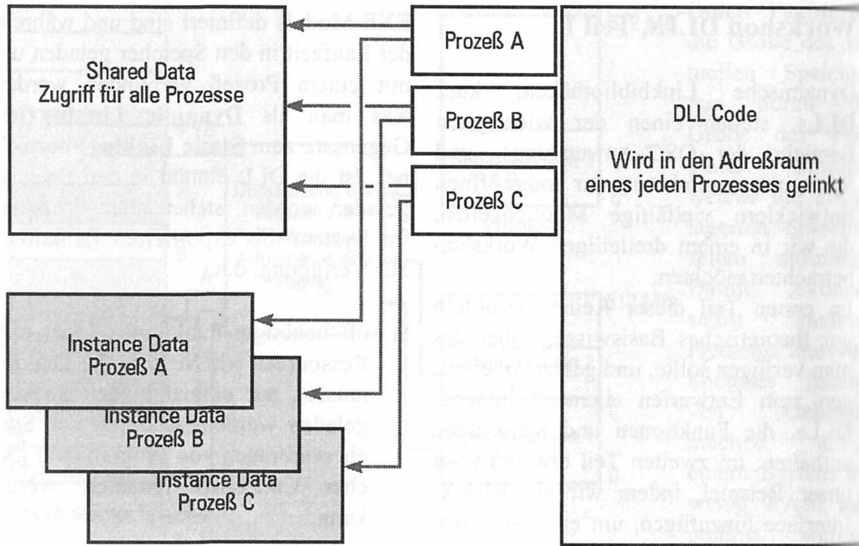


Abb.1: Verwendung von Instance und Shared Data durch verschiedene Prozesse.

zesses ausgeführt, der sie aufruft. So laufen z.B. Threads, die innerhalb einer DLL gestartet werden, im Prozeßraum des aufrufenden Prozesses und können auf die Daten innerhalb dieses Adreßraumes zugreifen. Der Code einer DLL wird also von allen Prozessen, die von ihr machen, geteilt; die Datenbereiche jedoch nicht unbedingt.
Grundsätzlich gibt es zwei verschiedene Arten von Daten, die eine DLL enthalten kann:

1. *Prozeßeigene Daten nennt man* **Instance Data**: Für jeden Prozeß, der eine DLL lädt, erstellt OS/2 eine Kopie der *Instance Data*, die sich im privaten Adreßraum des Prozesses befindet. Auf diese Weise wird vermieden, daß ein Prozeß Daten anderer Prozesse überschreibt, die ebenfalls die DLL benutzen.
2. *Für alle Prozesse zur Verfügung stehende Daten nennt man* **Shared Data**: Dies ist ein öffentlicher Datenbereich. Änderungen, die in Prozeß in diesem Bereich vornimmt, wirken sich auch auf alle anderen Prozesse aus, welche die DLL verwenden. Man kann so eine effiziente Methode der **Inter Process Communication (IPC)** entwickeln, was natürlich voraussetzt, daß der Zugriff auf den öffentlichen Speicherbereich der DLL entsprechend serialisiert wird.

Alternativ bietet sich die Möglichkeit, sowohl *Shared Data* als auch für jeden Prozeß private Speicherbereiche anzulegen (siehe Abb. 1).
Wie das System den Speicher beim Laden der DLL zuteilt, wird durch Attribute des Schlüsselwortes *DATA* in der DEF-Datei (**Module Definiton File**)

beim Linken der DLL festgelegt. Alle für DLLs wichtige Schlüsselworte mit ihren Attributen und ihren Bedeutungen finden Sie im Kasten *Relevante Schlüsselworte und Attribute für DLL DEF-Dateien*.

Initialisierung und Terminierung

Es kann manchmal nötig sein, daß bestimmte Aufgaben bei der Initialisierung respektive Terminierung einer DLL erledigt werden müssen, bevor Prozesse die Bibliothek verwenden bzw. sie freigeben können. So kann man z.B. bestimmte Standardwerte in den öffentlichen Datenbereich der DLL schreiben, oder Speicher allokieren, wenn man die DLL lädt; und Initialisierungsdaten sichern oder Speicher freigeben, wenn

Relevante Schlüsselworte und Attribute für DLL DEF-Dateien

LIBRARY

Syntax: LIBRARY <Name der DLL> <Initialisierung> <Terminierung>

Bedeutung:

Mit diesem Schlüsselwort definiert man den Namen der DLL und legt das Verhalten der Bibliothek bezüglich Initialisierung und Terminierung fest. Der Name der DLL wird dabei ohne die Dateierweitung *.DLL angegeben. Dem Namen folgen die unten angegebenen Attribute. Dabei gilt: Wird als <Initialisierung> INITINSTANCE angegeben, gilt automatisch TERMINSTANCE für <Terminierung>; wird als <Initialisierung> INITGLOBAL angegeben, gilt automatisch TERMGLOBAL.

Attribute:

INITINSTANCE

Die Initialisierungs/- Terminierungsfunktion *_DLL_InitTerm* wird zur Initialisierung immer dann aufgerufen, wenn ein Prozeß eine DLL lädt. Die Funktion läuft dabei innerhalb des Adreßraumes desjenigen Prozesses, der die DLL verwenden möchte.

INITGLOBAL

Die Initialisierungs/-Terminierungsfunktion *_DLL_InitTerm* wird zur Initialisierung nur einmal aufgerufen, und zwar wenn die DLL zum ersten Mal in den Speicher geladen wird. Die Funktion läuft dann zur Initialisierung innerhalb des Adreßbraumes desjenigen Prozesses, der die DLL zum ersten Mal lädt. Diese Einstellung ist standardmäßig aktiviert.

TERMINSTANCE

Die Initialisierungs/-Terminierungsfunktion *_DLL_InitTerm* wird zur Terminierung immer dann aufgerufen, wenn ein Prozeß eine DLL freigibt. Die Funktion läuft dabei innerhalb des Adreßraumes desjeningen Prozesses, der die DLL freigibt.

TERMGLOBAL

Die Initialisierungs/-Terminierungsfunktion *_DLL_InitTerm* wird zur Terminierung nur einmal aufgerufen, und zwar wenn die DLL aus dem Speicher entfernt wird. Die Funktion läuft dabei innerhalb des Adreßraumes desjenigen Prozesses, der die DLL als letzter freigibt. Diese Einstellung ist standardmäßig aktiviert.

Relevante Schlüsselworte und Attribute für DLL DEF-Dateien (Fortsetzung)

DATA

Syntax: DATA <Attribute...>

Bedeutung:
Das Schlüsselwort legt fest, wie auf Datenbereiche der DLL, die vom System beim Laden erzeugt werden, zugegriffen werden darf und wann sie in den Speicher geladen werden.

Attribute:
MULTIPLE
OS/2 erzeugt für jeden Prozeß, der eine DLL lädt, eine eigene Kopie der *Instance Data.* Nur der Prozeß, der die DLL geladen hat, kann auf diesen Datenbereich zugreifen. Diese Einstellung ist standardmäßig aktiviert. Sie kann mit dem Attribut NONSHARED kombiniert werden.

SINGLE
OS/2 erzeugt zum Zeitpunkt, zu dem die DLL das erste Mal geladen wird, einen einzigen Datenbereich, auf den alle Prozesse zugreifen können, welche die DLL später laden. Diese Einstellung kann mit dem Attribut SHARED kombiniert werden.

READWRITE
In die während des Ladens der DLL erstellten Datenbereiche kann geschrieben und aus ihnen gelesen werden. Das gilt sowohl für *Shared-* als auch für *Instance Data*. Diese Einstellung ist standardmäßig aktiviert.

READONLY
Aus den während des Ladens der DLL erstellen Datenbereiche kann nur gelesen werden. Das gilt sowohl für *Shared-* als auch für *Instance Data*.

LOADONCALL
Die Datenbereiche werden erst dann in den Speicher geladen, wenn sie tatsächlich benötigt werden. Das gilt sowohl für *Shared-* als auch für *Instance Data*. Diese Einstellung ist standardmäßig aktiviert.

PRELOAD
Die Datenbereiche sind sofort im Speicher verfügbar, nachdem ein Prozeß die DLL geladen hat. Das gilt sowohl für *Shared-* als auch *Instance Data*.

SEGMENTS

Syntax: SEGMENTS <[']Segmentname['] CLASS 'Klassenbezeichnung' [Attribute...]>

Bedeutung:
Mit diesem Schlüsselwort kann man einzelne Datenbereiche innerhalb der DLL definieren, welche nicht den Angaben entsprechen, die im Schlüsselwort DATA festgelegt wurden.

Relevante Schlüsselworte und Attribute für DLL DEF-Dateien (Fortsetzung)

Das ist notwendig, um DLLs zu erstellen, die sowohl *Shared-* als auch *Instance Data* aufweisen müssen. Für diesen Zweck ist als Klassenbezeichnung DATA anzugeben, da standardmäßig CODE gesetzt ist; als Attribut ist SHARED anzugeben, da standardmäßig NONSHARED gesetzt ist. Hinweis: Der Name des Segmentes muß in Hochkommata gesetzt werden, wenn der Name mit anderen Schlüsselworten wie CODE identisch ist.

CODE

Syntax: CODE <Attribut>

Bedeutung:
Legt fest, zu welchem Zeitpunkt der DLL-Code in den Speicher geladen werden soll.

Attribute:
LOADONCALL
Der DLL-Code wird erst dann in den Speicher geladen, wenn einer der Prozesse, der die DLL geladen hat, den Code benötigt. Diese Einstellung ist standardmäßig aktiviert.

PRELOAD
Der DLL-Code wird sofort in den Speicher geladen, wenn der erste Prozeß die DLL lädt.

EXPORTS

Syntax: EXPORTS
<Namen der zu exportierenden Funktionen>

Bedeutung:
Gibt die Namen der Funktionen an, die von der DLL exportiert werden sollen, damit Sie anderen Prozessen, welche die DLL laden, zur Verfügung stehen.

PROTMODE

Syntax: PROTMODE

Bedeutung:
Legt fest, ob die DLL nur im Protected Mode oder auch im Real Mode verwendet werden kann. Diese Angabe ist optional.

DESCRIPTION

Syntax: DESCRIPTION <'Beschreibung der DLL'>

Bedeutung:
Definiert eine Beschreibung der DLL. Diese Angabe ist optional.

die DLL aus dem Speicher entfernt wird. Für diese Aufgabe dient die Funktion *_DLL_InitTerm*, die vom System beim Laden und Freigeben der DLL aufgerufen wird. Die Funktion läuft dabei im Prozeßraum desjenigen Prozesses, der

die DLL lädt, bzw. der sie freigibt. Wie oft und zu welchen Zeitpunkten *_DLL_InitTerm* aufgerufen wird, legt man mit Hilfe der Attribute *INITGLOBAL* bzw. *INITINSTANCE* im *LIBRARY*-Schlüsselwort der DEF-Datei der DLL fest. Sie finden eine genaue Beschriebung dieser Attribute ebenfalls im Kasten *Relevante Schlüsselworte und Attribute für DLL DEF-Dateien.*

Um entsprechende Aufgaben zum Zeitpunkt des Ladens und Freigebens der DLL erledigen zu können, ist es erforderlich, eine eigene *_DLL_InitTerm* zu schreiben. Dies ist aber für die meisten DLLs nicht notwendig (wie für die in diesem Teil unserers Workshops vorgestellten Bibliotheken). Für komplexere Aufgaben ist das Entwickeln einer eigenen *_DLL_InitTerm*-Funktion aber erforderlich. Wir werden die Anwendung dieser Routine im dritten Teil des Workshops genau beschreiben.

DEF-Dateien für DLLs

Die Auflistung der Schlüsselworte und Attribute zeigt, daß man bereits vor dem Entwickeln der DLL genau wissen sollte, welche Aufgaben sie zu erfüllen hat, um den Umgang mit dem Speicher genau festzulegen. Im wesentlichen gibt es drei Anwendungsfälle:

1. DLLs nur mit *Instance Data*

   Handelt es sich darum, komplexe Funktionen, die von vielen Programmen aufgerufen werden sollen, in einer DLL zu codieren, erstellt man eine Bibliothek, welche die gewünschte Funktionalität umfaßt und die nötigen Funktionen exportiert. Sie sollte für jeden Prozeß einen eigenen Datenbereich zur Verfügung stellen. Der Entwurf einer Initialisierungsroutine wird für eine reine Funktionsbibliothek in den meisten Fällen nicht nötig sein.

2. DLLs, die keinen Code enthalten

   Möchte man Ressourcen für alle Prozesse zur Verfügung zu stellen, so trenne man sie stets vom Funktionscode und erzeuge eine DLL, die nur Ressourcen enthält. Wenn man zum Beispiel eine neue Fensterklasse schreibt, die Menüs und Dialoge benötigt, so definiere man für diese Ressourcen eine eigene DLL, aus der die von der Fensterklasse benötigen Ressourcen geladen werden.

3. DLLs mit *Instance* und *Shared Data*

   Sobald Systemressourcen im Spiel sind (etwa die Verwaltung einer Schnittstelle) oder falls man mit Datenbeständen arbeitet, die von vielen Prozessen nicht nur gelesen, sondern auch weiterverarbeitet werden, schreite man zur Entwicklung eines sogenannten Subsystems, welches in einer DLL enthalten sein kann. Dazu definiert man einen für jeden Prozeß privaten und einen öffentlichen, für alle Prozesse zugänglichen Datenbereich und erstellt eine Routine zur Initialisierung und Terminierung der DLL, die auch die Verwaltung der einzelnen Prozesse übernimmt. Besonders für umfangreiche Programmkonzepte bietet sich die Entwicklung einer Subsystem-DLL an.

Beispiele für DEF-Dateien zur DLL-Entwicklung

Beispiel einer DEF-Datei für eine DLL nur mit Instance Data

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; DEF-Datei für eine reine Funktionsbibliothek
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LIBRARY LIB01 INITINSTANCE TERMINSTANCE

PROTMODE
DATA MULTIPLE NONSHARED READWRITE LOADONCALL
CODE LOADONCALL

EXPORTS
    _BeispielFunktion

DESCRIPTION 'DLL zur Implementierung einer reinen Funktionsbibliothek'
```

Beispiel einer DEF-Datei für eine DLL, die nur Ressourcen enthält

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; DEF-Datei eine DLL, die nur Ressourcen enthält
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LIBRARY LIB02

PROTMODE
DESCRIPTION 'DLL zur Verfügungstellung von Ressourcen'
```

Beispiel einer DEF-Datei für eine DLL mit Shared und Instance Data

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; DEF-Datei für eine DLL mit Shared und  Instance Data
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
LIBRARY LIB03 INITINSTANCE TERMINSTANCE

PROTMODE
DATA MULTIPLE NONSHARED READWRITE LOADONCALL
CODE LOADONCALL

SEGMENTS
 Shared_Data CLASS 'DATA' SHARED

EXPORTS
    _BeispielFunktion

DESCRIPTION 'DLL zur Implementierung eines Subsystems'
```

Für diese drei Anwendungsfälle stellen wir Ihnen drei unterschiedliche DEF-Dateien zur Verfügung (siehe Kasten *Beispiele für DEF-Dateien zur DLL-Entwicklung*). Bis auf den Namen, den Sie Ihrer DLL geben, sind keinerlei Modifikationen erforderlich. Entsprechende Vorgaben für Code- und Header-Dateien entnehmen Sie bitte den einzelnen Beispielprogrammen unseres Workshops.

Wie dynamisches Linken funktioniert

Ein PM-Programm besteht zu großen Teilen aus Aufrufen von Funktionen mit dem Präfix *Win...*, die in der PMWIN.DLL definiert sind. Das Programm enthält in diesem Fall nicht den Code irgend einer *Win...*-Funktion (wie beim Static Linking), sondern an der Stelle des Funktionsaufrufes einen Pointer auf eine Adresse der Funktion, d.h. den Einsprungpunkt, an dem die Funktion in der DLL zu finden ist. Zum Zeitpunkt des Compilierens sind diese Adressen nur Dummyadressen, die während der Laufzeit des Programms aufgelöst werden. Dazu gibt es zwei Möglichkeiten:

1. Das Betriebssystem führt das Laden der DLL und die Auflösung der Adressen automatisch durch. Dabei wird während des Linkens des Programms eine Tabelle in der EXE-Datei angelegt, in welcher der Name der DLL und deren Funktionen aufgelistet sind. Beim Laden der EXE-Datei wird mit Hilfe der Tabelle bestimmt, welche DLL benötigt wird, der Code der DLL wird in den Adreßraum des Prozesses gelinkt, und die Dummyadressen werden durch die tatsächlichen Einsprungadressen (den sogenannten *Entry points*) in den DLLs ersetzt.
2. Das Programm übernimmt das Laden der DLL und die Ermittlung der Einsprungspunkte der Adressen. Dabei wird mit der Funktion *DosLoadModule* die DLL in den Speicher geladen. Mittels der Funkton *DosQueryAddrProc* wird die benötigte Adresse aus der DLL gelesen. Nur die Codeabschnitte der DLL werden in den Adreßraums des Prozesses gelinkt, die von der Anwendung geladen werden.

Die erste Methode ist die einfachste, weil man dem System die Arbeit überläßt. Dafür ist sie unflexibel: Zum einen muß das Verzeichnis, in dem die DLL liegt, in der LIBPATH-Anweisung eingetragen sein (nach der Installation muß das System also neu gestartet werden), oder die DLL wird in das gleiche Verzeichnis wie die EXE-Datei kopiert, weil das System zuerst dieses Verzeichnis, dann die im LIBPATH angegebenen Verzeichnisse nach der benötigten DLL absucht; zum anderen kann das Programm auf keinerlei Fehler reagieren, die während des Ladens einer DLL auftreten: Findet das System die Bibliothek nicht, kann das Programm einfach nicht gestartet werden. Der dritte und wohl schwerwiegendste Nachteil besteht allerdings darin, daß die DLL erst dann wieder aus dem Speicher entfernt wird, wenn der letzte Prozeß, der sie benötigt, beendet wird. Das kann zu unerwünschtem Speicherverbrauch führen, weil man einige DLLs nur

für bestimmte Zwecke kurze Zeit benötigt. Man sollte als Entwickler stets so sparsam wie nur irgend möglich mit dem Speicher umgehen und nicht der Ansicht sein, moderne Hardware kompensiere ein schwerfälliges Softwaredesign (was leider bei »moderner« Software eine Art Maxime ist). Wir stellen Ihnen daher bewußt nur die zweite Methode vor.

Code für das EXE-Modul

Zu welchem Zeitpunkt soll eine DLL nun geladen werden, und wie sind die bereits erwähnten Funktionen einzusetzen?

Bevor man eine DLL in einem Programm verwenden möchte, muß man sie mit der Funktion *DosLoadModule* laden. Diese Funktion gibt ein Handle vom Typ HMODULE auf die DLL zurück, welches von verschiedenen Funktionen benötigt wird und in einer Variablen gespeichert wird. Der Funktionsaufruf kann grundsätzlich von jeder Stelle in einem Programm erfolgen auch in Unterprogrammen oder einer Fensteprozedur stehen. Beachten Sie aber, die Funktion für eine DLL innerhalb eines EXE-Moduls nur einmal aufzurufen. Eine Abfrage wie:

```
if (!hmodLIB01)
  rc = DosLoadModule (NULL, 0,
                      "LIB01",
                      &hmodLIB01);
```

ist oftmals die beste Lösung. Daneben beachten Sie bitte die folgenden Tips:

- Innerhalb einer Fensterprozedur rufen Sie *DosLoadModule* am besten innerhalb des Anweisungsblocks für die WM_CREATE-Verarbeitung oder einer von Ihnen selbst definierten Nachricht auf, die nur einmal zum Zeitpunkt der Initialisierung des Fensters abgearbeitet wird.

- Die Variable mit dem DLL-Handle können Sie natürlich auch als Parameter weitergeben. So wäre es möglich, den Handle auf eine Ressourcen-DLL im Feld einer Initialisierungsstruktur zu speichern und einen Pointer auf diese Struktur mit einer Nachricht an neu erstellte Fenster zu schicken. Mit dem Handle können die Fenster Ressourcen aus der DLL laden und verwenden. Dabei ist es natürlich auch möglich, daß die Fensterprozedur ebenfalls in einer DLL enthalten ist.

- Fensterklassen, die in DLLs enthalten sind, registieren Sie vom EXE-Modul aus, am einfachsten mit einer DLL-Funktion zur Klassenregistrierung, welche von der DLL exportiert wird. Die Verarbeitung von DLL-Fensterklassen zeigt der Beispielcode für diesen Teil des Workshops.

- Fensterklassenspezifische Funktionen (wie Berechnungsmethoden) definieren Sie zusammen mit der Fensterprozedur dieser Klasse in der gleichen DLL.

Die Funktion *DosLoadModule* hat folgende Syntax:

```
rc = DosLoadModule (ObjNameBuf
                    ObjNameBufL
                    ModName
                    ModHandle);
```

RC	*Konstante*	*Bedeutung*
0	NO_ERROR	Fehlerfreier Ablauf
2	ERROR_FILE_NOT_FOUND	Die DLL konnte nicht gefunden werden. *Ihr Name wurde falsch angegeben, sie existiert nicht, oder sie ist nicht im angegebenen Verzeichnis.*
3	ERROR_PATH_NOT FOUND	Der angegebene Pfad konnte nicht gefunden werden. *Er wurde falsch angegeben oder er existiert nicht.*
4	ERROR_TOO_MANY_OPEN_FILES	Keine weitere Datei kann geöffnet werden. *Der Prozeß hat bereits zuviele Dateien geöffnet.*
5	ERROR_ACCESS_DENIED	Zugriff auf die Datei verweigert.
8	ERROR_NOT_ENOUGH_MEMORY	Es steht nicht genug Speicher zur Verfügung.
11	ERROR_BAD_FORMAT	Das Format der DLL ist fehlerhaft. *DLL neu compilieren und linken.*
26	ERROR_NOT_DOS_DISK	Auf das Laufwerk kann nicht zugegriffen werden. *Laufwerk überprüfen und ggfs. formatieren.*
32	ERROR_SHARING_VIOLATION	Auf die DLL konnte nicht zugegriffen werden. *Erneut versuchen, das Programm zu starten.*
33	ERROR_LOCK_VIOLATION	Die Datei ist für einen Zugriff gesperrt.
36	ERROR_SHARING_BUFFER_EXCEEDED	Um die Datei zu öffnen und zu bearbeiten, ist nicht genügend Speicher vorhanden.
95	ERROR_INTERRUPT	Es trat ein interner Fehler von *DosLoadModule* auf. *Erneut versuchen, das Programm zu starten.*
108	ERROR_DRIVE_LOCKED	Das Laufwerk ist für einen Zugriff gesperrt.
123	ERROR_INVALID_NAME	Der angegebene Name der DLL ist nicht korrekt. *Der Name muß den HPFS-Regeln entsprechen.*
127	ERROR_PROC_NOT_FOUND	Die Funktion *_DLL_InitTerm* - Funktion wurde nicht gefunden. *DLL-Code überprüfen.*
180	ERROR_INVALID_SEGMENT_NUMBER	Der Aufbau der DLL ist fehlerhaft *(s. rc = 11).*
182	ERROR_INVALID_ORDINAL	Das System fand die Ordinalnummer einer Funktion nicht. *Das gesamte Programmprojekt mit der aktuellen Import-Bibliothek neu erstellen.*
190	ERROR_INVALID_MODULETYPE	Der Aufbau der DLL ist fehlerhaft *(s. rc = 11).*
191	ERROR_INVALID_EXE_SIGNATURE	Die DLL ist keine OS/2-DLL
192	ERROR_EXE_MARKED_INVALID	Der Aufbau der DLL ist fehlerhaft *(s. rc = 11).*
194	ERROR_INTERATED_DATA_EXCEEDS_64K	Das System hat Schwierigkeiten, Speicher für die DLL bereitzustellen *(s. rc = 11).*
195	ERROR_INVALID_MINALLOCSIZE	s. rc = 194
196	ERROR_DYNLINK_FROM_INVALID_RING	Der Prozeß, der die DLL laden möchte, läuft nicht auf der gleichen Privilegebene. *PL überprüfen.*
198	ERROR_INVALID_SEGDPL	s. rc = 194
199	ERROR_AUTODATASEG_EXCEEDS_64K	s. rc = 194
201	ERROR_RELOCSCR_CHAIN_EXCEEDS_SEGLIMIT	s. rc = 194
206	ERROR_FILENAME_EXCED_RANGE	Der Dateiname ist zu lang. *HPFS-Regeln beachten.*
295	ERROR_INIT_ROUTINE_FAILED	*_DLL_InitTerm* gibt 0 zurück. *Funktion prüfen.*

Tabelle 1: Rückgabewerte von *DosLoadModule,* ihre Bedeutungen und Lösungstips

Dabei zeigt *ObjNameBuf* auf eine Zeichenkette, die den Namen eines Objektes aufnehmen kann, das beim Laden einer DLL einen Fehler verursachte. Ist z.B. die DLL defekt, steht hier der Name der DLL. *ObjNameBufL* ist die Länge der zur Verfügung stehenden Zeichenkette. *ModName* gibt den Namen der DLL an, die geladen werden soll. Wird hier nur der Name der DLL angegeben, nimmt OS/2 an, sie befindet sich im aktuellen Verzeichnis (das, in dem auch das EXE-Modul liegt) oder im Verzeichnis DLL. Soll die DLL in einem anderen Verzeichnis plaziert werden, muß der vollständige Verzeichnisname angegeben werden. *ModHandle* enthält als Ausgabewert den Handle auf die DLL, sofern die Funktion erfolgreich war.

Wir werden die beiden ersten Parameter von *DosLoadModule* ignorieren. Allerdings werden wir genau auf den Rückgabewert achten. Es gibt insgesamt 28 Rückgabewerte dieser Funktion, die Sie samt ihrer Bedeutung in Tabelle 1 finden. Die wenigsten dieser Fehler begegnen dem Programmierer in der Praxis. Eigentlich hat man nur mit den Rückgabewerten 2 und 3 zu tun und kann die Fehler rasch abstellen, indem man die DLL in das erforderliche Verzeichnis kopiert oder die Angabe im Parameter *ModName* überprüft.

Ist die DLL geladen, fragt man die Adressen der benötigten Funktionen ab. In unseren Beispielen für DEF-Dateien haben wir die zu exportierenden Funktionen nur mit ihrem Namen aufgelistet, was übersichtlicher ist als die Auflistung mit Ordinalnummern. Um die Adressen der Funktionen abzufragen, verwendet man die Funktion *DosQueryProcAddr*, die einen Zeiger auf die Adresse der Funktion zurückliefert. Die Syntax der Funktion lautet:

```
rc = DosQueryProcAddr
        (HMODUDLE ModHandle,
         ULONG Ordinal,
         PSZ ProcName,
         PFN ProcAddr);
```

Hierin ist *ModHandle* der Handle auf die DLL, der mit *DosLoadModule* ermittelt wurde. *Ordinal* bezeichnet die Ordinalnummer der Funktion innerhalb der DLL. Ist Ordinal nicht 0, wird *ProcName* irgnoriert. *ProcName* bezeichnet den Namen der Funktion in der DLL. Die Variable *ProcAddr* enthält als Ausgabewert die Adresse der Funktion.

Wie bereits gesagt, geben wir den Namen der Funktion in der *EXPORTS*-Anweisung an, übergeben daher der Funktion *DosQueryProcAddr* den Funktionsnamen im Parameter *ProcName* und setzen *Ordinal* auf 0L. Bezüglich des Rückgabewertes definieren wir für jede Funktion, die wir aus der DLL importieren möchten, eine Variable vom Typ *PFN*, der wir den Namen der Funktion geben, z.B.:

```
PFN XeliaRegXeliaPushBtnClass;
```

Diese Variable verwendet man dann ganz normal als Funktion im Programmcode, so als hätte man die Funktion im EXE-Modul definiert, etwa:

```
rc = XeliaRegXeliaPushBtnClass (hab);
```

RC	Konstante	Bedeutung
0	NO_ERROR	Fehlerfreier Ablauf.
6	ERROR_INVALID_HANDLE	Der Handle, welcher der Funktion übergeben wurde, ist nicht gültig. *Sicherstellen, daß zuvor DosLoadModule fehlerfrei aufgerufen wurde.*
123	ERROR_INVALID_NAME	In der DLL gibt es keine Funktion mit der angegebenen Ordinalnummer oder dem angegebenen Namen. *Sicherstellen, daß der Name korrekt übergeben, bzw. eine Ordnialnummer angegeben wurde, die in der DLL auch existiert.*
65079	ERROR_ENTRY_IS_CALLGATE	Der Zugriff auf den ermittelten Einsprungpunkt kann nicht gewährt werden. *Auf den benötigten Einsprungpunkt kann nur über ein Call-Gate zugegriffen werden. Nicht mehr versuchen, auf diesen Einsprungpunkt zuzugreifen.*

Tabelle 2: Rückgabewerte von *DosQueryProcAddr,* ihre Bedeutungen und Lösungstips

Beachten Sie im Zusammenhang mit dieser Funktion folgende Punkte:

- Der im Parameter *ProcName* angegebene Name der Funktion muß exakt dem Namen in der EXPORTS-Anweisung entsprechen.
- Die PFN-Variable, die man im Parameter *ProcAdr* angibt, kann beliebig gewählt werden. Zweckmäßigerweise übernimmt man hier den wirklichen Namen der Funktion, deren Adresse man ermittelt.
- Die einzelnen PFN-Anweisungen sollten als *static* oder global definiert werden.
- Die Variablen können genauso als Parameter anderen Funktionen übergeben werden, wie der Handle auf die DLL. So wäre es auch hier denkbar, Funktionspointer in geeigneten Strukturen zu speichern und einen Zeiger auf diese Struktur mit einer Nachricht an ein Fenster zu schicken.

DosQueryProcAddr hat nur vier Rückgabewerte (Tab. 2), auf die wir ebenfalls reagieren werden. Fehler 123 ist hierbei ein Problem, das eigentlich nur während der Entwicklungsarbeit auftritt, wenn man den Namen einer Funktion versehentlich falsch angegeben hat. Fehler 65079 dürfte bei DLLs, die sich ausschließlich auf das OS/2 API stützen, nicht auftreten, und Fehler 6 müßte durch eine ordentliche Behandlung der möglichen Rückgabecodes von *DosLoadModule* vermieden werden können.

Damit Sie nicht soviel Arbeit mit der Bearbeitung der Fehlercodes haben, finden Sie im Beispielcode eine Funktion namens *ProcessDLLImpErrors.* Diese Funktion bearbeitet die Fehler beider API-Funktionen, gibt entsprechende Fehlermeldungen aus, wenn Fehler nicht behebbar sind, und ermöglicht eine besondere Bearbeitung der *DosLoadModule-RCs* 2 und 3, indem nach der DLL gesucht wird, wenn sie im angegebenen Verzeichnis nicht gefunden werden kann. Anschließend wird versucht, die DLL

erneut zu laden, um das Programm fortsetzen zu können. Eine genaue Beschreibung von *ProcessDLLImpErrors* finden Sie in der Readmedatei zu den Beispielprogrammen.

Die dritte API-Funktion, die im EXE-Modul aufgerufen werden muß, heißt *DosFreeModule* und dient zum Freigeben der vorerst geladenen DLL. Die Funktion hat folgende Syntax:

rc = DosFreeModule
(HMODULE ModHandle);

Dabei ist *ModHandle* der zuvor ermittelte Handle auf die DLL.
Diese Funktion können Sie an jeder Stelle Ihres Programmes aufrufen.

! ***Nachdem* DosFreeModule *aufgerufen wurde, ist der Handle auf die DLL nicht mehr länger gültig, und auch die Adressen in den PFN-Variablen nicht mehr! Versuchen Sie dann auf eine solche Funktion zuzugreifen, wird Ihr Programm beendet. Rufen Sie* DosFreeModule *also erst dann auf, wenn Sie sicher sind, daß Sie eine DLL nicht mehr brauchen.***

Beachten Sie außerdem folgende Hinweise:

- Rufen Sie *DosFreeModule* in Fensteprozeduren am besten im Anweisungsblock zur Bearbeitung der WM_DESTROY auf, aber nur dann, wenn zuvor *DosLoadModule* aufgerufen wurde.
- Achten Sie darauf, daß ein Kindfenster keine DLL freigibt, die noch vom Hauptfenster benötigt wird. Überlassen Sie in PM-Programmen daher am besten immer dem Programmfenster bzw. dem Hauptprogramm das Laden und Freigeben für das gesamte Programm essentiell wichtiger DLLs.

Die Rückgabecodes von *DosFreeModule* entnehmen Sie bitte Tabelle 3. Wie der Gebrauch der drei Funktionen und unseres Programms *ProcessDLLImpErrors* in der Praxis aussieht, können Sie aus dem Codeabschnitt 1 ersehen und in allen Einzelheiten dem Beispielcode entnehmen.

RC	***Konstante***	***Bedeutung***
0	NO_ERROR	Fehlerfreier Ablauf.
6	ERROR_INVALID_HANDLE	Der Handle, welcher der Funktion übergeben wurde, ist nicht gültig. *Sicherstellen, daß zuvor DosLoadModule fehlerfrei aufgerufen wurde.*
12	ERROR_INVALID_ACCESS	*DosFreeModule* konnte auf die DLL nicht zugreifen, um die Bibliothek freizugeben. *Die Funktion noch einmal aufrufen. Sicherstellen, daß die DLL zuvor fehlerfrei geladen wurde.*
95	ERROR_INTERRUPT	Interner Fehler der Funktion *DosFreeModule. Die Funtktion noch einmal aufrufen.*

Tabelle 3: Rückgabewerte von *DosFreeModule,* ihre Bedeutungen und Lösungstips

Codebespiel 1: Anwendungsbeispiel der vorgestellten Funktionen

```
#define INCL_WIN
#define INCL_GPI
#define INCL_DOS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <os2.h>
#include "xelia.h"

// Globale Variablen
HMODULE hmodXeliaCtrlsDLL; // Module-Handle auf XCTRLS.DLL
PFN XeliaRegisterXeliaPushButtonClass;  // Function Pointers

INT main (VOID)
{
 HAB habAnchor;
 APIRET rc;
 // ... weitere Deklarationen

 // DLL XCTRLS laden
 rc= DosLoadModule(NULL, 0L,
                   "XCTRLS",        // DLL im gleichen Verzeichnis wie EXE
                   &hmodXeliaCtrlsDLL);
 if (rc!=0)
  ProcessDLLImpErrors (rc, "XCTRLS", &hmodXeliaCtrlsDLL);

 // Adresse von XeliaRegisterXeliaPushButtonClass abfragen
 DosQueryProcAddr (hmodXeliaCtrlsDLL, 0,
                   "_XeliaRegisterXeliaPushButtonClass",
                   &XeliaRegisterXeliaPushButtonClass);
 if (rc!=0)
  ProcessDLLImpErrors (rc, "XCTRLS", &hmodXeliaCtrlsDLL);

 //...

 XeliaRegisterXeliaPushButtonClass (habAnchor);

 //...


 // DLL XCTRLS freigeben
 DosFreeModule (hmodXeliaCtrlsDLL);
}
```

Wir laden zunächst die DLL mit *DosLoadModule* und geben in Parameter 3 als Modulbezeichnung nur den Namen der DLL *ohne die Dateierweiterung* an. Damit wird vorausgesetzt, daß sich die DLL im gleichen Verzeichnis wie die Programmdatei befindet. Möchte man die DLL in einem gesonderten Verzeichnis ablegen, ist hier die volle Pfadangabe (wieder ohne die Dateierweiterung) anzugeben, etwa "C:\\DLLS\\XCTRLS", wobei jeder Backslash stets zweimal angegeben wird. Den Namen der DLL wird man natürlich nur dann direkt als Parameter im Code angeben, wenn die Bibliothek tatsächlich im gleichen Verzeichnis wie die EXE-Datei liegen darf. Eleganter geht es mit einer INI-Datei, in der man die Pfade aller vom Programm benötigter DLLs speichern und sie vor dem Laden abfragen kann.
Verläuft der Ladevorgang der DLL nicht erfolgreich, übergeben wir den Fehlercode der Funktion *ProcessDLLImpErrors*, die auf den Mißerfolg des Vorgangs reagiert. Sollte die DLL nicht gefunden werden, durchsucht die Funktion daraufhin das System nach der DLL. Nachdem die Fehlerquelle abgestellt ist, lädt *ProcessDLLImpErrors* die DLL noch einmal. Der gültige Handle auf die DLL wird in der Variable *hmodXeliaCtrlsDLL* gespeichert. Anschließend wird die Adresse der Funktion *_XeliaRegisterXeliaPushButtonClass* mit *DosQueryProcAddr* abgefragt und in der Variable *XeliaRegisterXeliaPushButtonClass* gespeichert, die als PFN deklariert ist. Sollte das Abfragen der Adresse nicht erfolgreich sein, übergibt das Programm der Funktion *ProcessDLLImpErrors* den Fehlercode, die darauf entsprechend reagiert. Zum Schluß des Programms geben wir die DLL mit *DosFreeModule* wieder frei.

Code für die DLL

Für den DLL Code gibt es nur einige Dinge zu beachten, die wir kurz zusammenfassen wollen:

- Funktionen, die von der DLL exportiert werden, müssen im Prototyp mit dem Schlüsselwort *APIENTRY* versehen werden (s. Codebeispiel 2).
- Den Funktionsnamen stellt man in der DLL am besten ein _ voran, wie bei *_XeliaSetStatusbarText*. Im EXE-Modul nennt man die importierte Funktion dann *XeliaSetStatusbarText*. Diese Maßnahme trägt der Übersicht bei.
- Alle Variablen, die in der DLL als global definiert werden, gehören zur *Instance Data* und sind daher prozeßspezifisch.

Ein neues Control in einer DLL

Mit unserem Beispielcode zu diesem Teil des Workshops befassen wir uns mit dem ersten der drei Anwendungsfälle einer DLL. Dazu entwicklen wir einen neuen Pushbutton, der in einer Statuszeile einen Hilfetext anzeigt, immer wenn sich der Mauszeiger über ihm befindet. Zusätzlich soll es möglich sein, daß der neue Pushbutton gleichzeitig die Aufgaben einer Statuszeile übernehmen kann. Da es sich bei einem Control um ein Fenster handelt, wird eine neue Fensterklasse geschrieben, welche in der DLL enthal-

Codebeispiel 2: Header -Datei für eine DLL

```
//-----------------------------------------
// Definitonen für Xelia Push Buttons
//-----------------------------------------
// Fensterklassen
#define WC_XELIAPUSHBUTTON "XeliaControlCenterButtonControlClass"

// Eigene Nachrichten
#define XELIA_BTN_SETBUTTONHELPTEXT (WM_USER+2)
#define XELIA_BTN_SETSTDHELPTEXT (WM_USER+3)

// Typendeklarationen
typedef struct
{
  USHORT cb;
  PSZ  pszText ;
  BOOL fHaveCapture;
  BOOL fHaveFocus;
  BOOL fInsideRect;
  BOOL bIsStatusbar;
}
XPUSHBTN, *PXPUSHBTN;

// Exportierte Funktionen
BOOL APIENTRY _XeliaRegisterXeliaPushButtonClass (HAB hab);
HWND APIENTRY _XeliaCreateXeliaPushButton (HWND hwndParent,
                                           HWND hwndOwner,
                                           PSZ pszBtnText,
                                           LONG lx, LONG ly,
                                           LONG lcx, LONG lcy,
                                           LONG lBtnID, BOOL bIsStatusbar);
BOOL APIENTRY _XeliaSetStatusbarText (PSZ pszStatusString);

// Interne Funktionen
MRESULT EXPENTRY XeliaPushButtonProcedure (HWND hwndWnd,
                                           ULONG ulMsg,
                                           MPARAM mpParam1,
                                           MPARAM mpParam2);

// Dateiende
```

ten ist. Für den Umgang mit dem neuen Kontrollelement fügen wir noch einige Funktionen hinzu, die man sowohl innerhalb als auch außerhalb der DLL aufrufen kann. Ein einfaches Beispielprogramm lädt die DLL und erzeugt drei der neuen Buttons, um deren Funktion zu überprüfen. Das Codebeispiel für die

DLL ist eine stark gekürzte Fassung aus den Dateien unserer Architektur Xelia.

Den Beispielcode besorgen Sie sich am besten von unserer Homepage. Ist Ihnen das nicht möglich, können Sie Ihn auch auf Diskette über uns beziehen.

Wenn Sie den Code durchgehen, werden Sie die wichtigsten Fragen, die auftauchen können, bereits geklärt wissen. Folgen Sie außerdem den Kommentaren im Code und der Readmedatei zum Programmpaket. Es sollte damit problemlos sein, den Beispielcode zu verstehen.

Ressourcen-DLLs

Die Hilfefunktion des neuen Buttons ist eine feine Sache. Unangenehm ist aber, daß die Hilfetexte mit dem EXE-Modul verbunden sind. Möchte man nun Hilfe in einer anderen Sprache bieten, müßte man die Ressourcendatei des Executables verändern und das gesamte Modul neu compilieren. Demnach wäre es wünschenswert, die Ressourcen vom Programm zu trennen. Wir werden dies tun und für die zweite Version unseres kleinen Programms zwei neue DLLs schreiben, die nur Ressourcen enthalten, einmal in Deutsch und einmal in Englisch. Hier sieht man den Vorteil einer DLL: Ein Teil des Projektes (in diesem Falle der Code) bleibt von der weiteren Entwicklungsarbeit völlig unberührt, während man einen anderen Teil separat verändern kann, was Zeit und Mühen spart. Im zweiten Teil des Workshops wird klar, wie durch die Weiterentwicklung der Code-DLL die Funktionalität des EXE-Moduls steigt, ohne daß Veränderungen daran vorgenommen werden müßten.

Um zu entscheiden, in welcher Sprache das Programm laufen soll, zeigt es zu Beginn einen Dialog an, in dem man die zu verwendende Sprache festlegen kann. Anschließend lädt es aufgrund der Dialoginformationen die entsprechende Ressourcen-DLL und lädt bei Bedarf die nötigen Strings aus der DLL, um sie in der Statuszeile anzeigen zu lassen.

Ressourcen binden Sie mit Hilfe einer RC-Datei genauso in eine DLL ein wie in ein Executable. Obwohl Ihre DLL nur Ressourcen enthalten soll, muß der Compiler eine Datei vorfinden. Also erstellen Sie eine Dummy-Code-Datei, die den Namen der DLL trägt und lassen diese zusammen mit den Ressourcen compilieren und linken. Das war schon alles. Der Code der RC-Dateien sollte beinahe selbsterklärend sein. Beachten müssen Sie nur, daß Sie die IDs, welche in der RC-Datei verwendet werden, auch in alle EXEs einbinden, da Ressourcen über ihre IDs identifiziert werden. Man legt dazu am besten eine Headerdatei an, die sämtliche IDs der Ressourcen enthält, womit man die Deklarationen bequem in neue Programme einbinden kann.

Die Ressourcen-DLL wird genauso geladen wie die DLL mit den Funktionen für den neuen Pushbutton. Adressen fragt man natürlich keine ab. Die Ressourcen selbst lädt man mit dem Module-Handle, den man nach dem erfolgreichen Laden von *DosLoadModule* erhalten hat (spezifiziert die DLL), und mit der entsprechenden ID (spezifiziert den Eintrag in der DLL). Unser Beispielprogramm benötigt nur Hilfetexte, die in einer Stringtable in der DLL gespeichert sind. Um die Texte zu laden, verwendet das

Programm die Funktion *WinLoadString*. Die Syntax lautet:

lLength = WinLoadString
(HAB hab,
HMODULE Resource,
ULONG idString,
ULONG lBufferMax,
PSZ pszBuffer);

hab ist dabei der Anchor Block Handle der Anwendung. Unser Beispielprogramm fragt ihn mit *WinQueryAnchorBlock* ab. *Resource* ist der Module Handle der DLL, welche die Ressourcen beinhaltet. *idString* bezeichnet die ID des Strings, der in einer Stringtable gespeichert ist. *lBufferMax* gibt die Größe des Ausgabepuffers *pszBuffer* an, in den die geladene Zeichenkette gespeichert werden soll. Als Rückgabewert gibt die Funktion die Länge des geladenen Strings in *lLength* zurück.
Der String wird mit seiner ID aus der DLL geladen und in der Variable *pszStatusString* gespeichert. Dann wird er mit der DLL-Funktion *XeliaSetStatusbarText* an die Fensterklasse des neuen Pushbuttons in der DLL geschickt und dort so verarbeitet, daß er in unserer Statuszeile angezeigt wird. Es gibt noch mehr API-Funktionen, um Ressourcen aus einer DLL zu laden. Diese sind:

WinLoadAccelTable
Zum Laden von Accelerator-Tabellen

WinLoadDlg
Zum Laden von Dialog Templates

WinLoadHelpTable
Tabellen für die Online-Hilfefunktion

WinLoadMenu
Zum Laden eines Menüs

WinLoadMessage
Zum Laden einer Nachricht

WinLoadPointer
Zum Laden von Bitmaps und Zeigern.

Den Module-Handle wird auch in anderen Funktionen angegeben, etwa beim Aufruf von *WinCreateStdWindow*. Sie können das Programmicon, Menüs und Accelerator-Tabellen ebenfalls in einer DLL speichern, sie laden und dann das Hauptfenster erstellen. Die Funktion *WinCreateStdWindow* lädt die Ressourcen automatisch aus der DLL. Bedingung ist, daß die zu ladenden Ressorcen die ID des Hauptfensters tragen. Beachten Sie hierzu unseren Beispielcode.

Zum Schluß

Den neuen Button kann man, wie das Standard-Control auch, mit beliebigen Funktionen belegen (in unserem Falle nur eine einfache Messagebox). Viel interessanter wäre es natürlich, wenn der Benutzer festlegen könnte, welche Funktionen durch das Programm auszuführen sind, wenn der Button betätigt wird. Man nennt solche benutzerdefinierten Programmabläufe Makros, und unter OS/2 lassen sich diese am besten mit REXX realisieren. Daher werden wir unsere DLL im nächsten Teil des Workshops mit einem REXX-Interface für den Makrosupport erweitern und außerdem am Beispiel einiger Funktionen für astronomische Berechnungen zeigen, wie man C-DLLs schreibt, die Funktionsbibliotheken für REXX-Skripte beinhalten. ❑

HTML Workshop, Teil 1

Im Web präsent zu sein, ist für viele Bedingung und gehört heutzutage zum guten Ton. Für viele Neueinsteiger ist die Erstellung eigener Webseiten aber ein Buch mit sieben Siegeln, und für sie ist es abschreckend, wenn sie den Quelltext eines *HTML*-Dokumentes betrachten. Wir möchten Ihnen daher in einem zweiteiligen Workshop *HTML* nahebringen und Ihnen neben einer Sprachreferenz mit wichtigen Tips den Weg in das Design eigener WWW-Seiten ebnen.
Im ersten Teil unseres Workshops lernen Sie grundsätzliche Dinge und Definitionen rund um *HTML* kennen und einige Regeln, die man beim Webdesign beachten sollte. Daneben möchten wir Ihnen in diesem Teil eine kleine *HTML*-Sprachreferenz in die Hand geben, die Sie zum Nachschlagen benutzen können, so daß Sie nicht immer das Web durchforsten müssen, wenn Sie mal schnell Informationen zu einem *Tag* brauchen. Neben Hinweisen zu den einzelnen Sprachelementen finden Sie auch Gerüste zum schnellen Design Ihrer Webseiten. Der zweite Teil des Workshops informiert Sie über Hilfsmittel wie *HTML*-Editoren, die Sie bei der Erstellung von *HTML*-Dokumenten unterstützen.

Wozu noch die Sprache kennen?

Das mögen sich manche fragen, gibt es doch WYSIWYG-*HTML*-Editoren, die eine genaue Kenntnis der Sprache scheinbar unnötig machen. Aber das ist keineswegs so. Visuell mit der Maus eine Webseite zu erstellen, ist natürlich eine feine Sache und sehr komfortabel; aber wer *HTML* kann, der im wesentlichen beherrscht auch die IPF-Sprache, mit der INF-Dokumente und Hilfedateien für OS/2 geschrieben werden können. Und grundsätzlich gilt, daß ohne eine Kenntnis von *HTML* das Design eigener Webdokumente nicht viel bringt. Wenn Sie in der Lage sind, nur mit dem Systemeditor Ihre Webseiten zu kreieren, kann Ihnen gar nichts passieren, wenn Sie mal selbst Änderungen an vorgefertigten Dokumenten vornehmen müssen, und es ist immer wichtig zu wissen, wie das, was man sieht, eigentlich aufgebaut ist, und was im Hintergrund eines Programms geschieht.

Was ist HTML?

HTML ist eine Abkürzung und steht für **Hyper Text Markup Language**. Die Sprache stellt einen Satz mehrerer, standardisierter und plattformunabhängiger Sprachelemente zur Verfügung, die man *Tags* nennt. Ursprünglich findet *HTML* seinen Ursprung als Eigentwicklung des *CERN*, dem Europäischen Institut für Kernphysik in der Schweiz, und wurde entworfen, um wissenschaftliche Ergebnisse in einer einfach und schnell einsehbaren und vor allem sauberen Form im Internet auszutauschen.
Ein *HTML*-Dokument ist eine ASCII-Textdatei, die mit jedem einfachen Editor erstellt werden kann. Um sie zu betrachten, benutzt man einen Browser, der die einzelnen *Tags* liest und entsprechend darstellt. Hier trifft man bei der Erstellung eigener Webseiten bereits auf die ersten Probleme: Browser stellen verschiedene *Tags*, bzw. deren Kombinationen, meist unterschiedlich dar, und

obwohl *HTML* standardisiert wurde, können einige Browser mit manchen Standards nicht umgehen. Bekannt dürften die *HTML*-Standards 2.0 und 3.0, sowie 3.2 sein, wenngleich es immer neue Versionen gibt, bereits 4.0 und höher. Der Sprachumfang wird dabei immer größer. Die Probleme, die damit auftreten, betrachten wir gleich.
Eine besondere Eigenschaft von *HTML*-Dokumenten ist die Möglichkeit, Links einzusetzen, um von einer Seite auf andere Seiten, Server, Dateien usw. zu verweisen. Man kann mit dem Browser Dateien via *FTP (FileTransfer-Protocol)* auf die eigene Maschine laden und auch andere Netzdienste, wie *Telnet* nutzen. Für Webdokumente gibt es übrigens auch ein eigenes Protokoll, das *HyperTextTransferProtokoll (HTTP)*, das die Übertragung von *HTML*-Dokumenten regelt. Man kennt die Bezeichnung *http* von den Adressen der Webserver. Die zahlreichen Möglichkeiten, die sich mit Links in einem *HTML*-Dokument eröffnen, erweitern ein Dokument, das zunächst nur Informationen in Textform bietet, zu einem interaktiven Einrichtung, da über die Links auch Programme angegeben werden können, die vom Server, auf dem das Dokument liegt, ausgeführt werden. Mit der Programmierung solcher Programme (CGI Skripte) werden wir uns auch noch befassen, wenn wir *HTML* beherrschen.

Was braucht man zum Webdesign?

Sie benötigen neben einem einfachen Texteditor (dem *Systemeditor* oder dem *Erweiterten Editor*), einen Webbrowser, um Ihre Dokumente auch betrachten zu können. Sie schreiben Ihr Dokument mit dem Editor, speichern es ab und laden es in den Browser, um die neuen Seiten zu kontrollieren.
Unter OS/2 bieten sich zwei Browser an, der *IBM WebExplorer* und natürlich das von Unix und Windows wohlbekannte *Netscape*. Das Programm gibt es seit Warp 4 auch für OS/2, und mittlerweile steht auch der *Netscape Communicater 4.0* zur Verfügung. Ein Programm, das eigentlich alles bietet, was man fürs Internet braucht. Wir werden uns damit in einer der nächsten Ausgaben intensiver beschäftigen.
Wir empfehlen Ihnen aber, beim Design Ihrer Webseiten am besten den *IBM WebExplorer* in seiner letzten Version 1.03 zu verwenden (unterstützt *HTML 3.0*). Der Grund liegt darin, daß sehr viele OS/2 Anwender trotz der Verfügbarkeit von *Netscape* den *WebExplorer* immer noch gerne verwenden. Das verwundert nicht sehr, der *WebExplorer* ist ein schnelles, kleines Programm; im Gegensatz zum etwas behäbigen Programmonster *Netscape*. Auch wir benutzen den *WebExplorer*. Leider wird er schon lange nicht mehr weiterentwickelt und unterstützt daher auch keine Frames, und andere *HTML*-Features die man im Web immer häufiger findet. Das führt uns gleich zu der Frage, ob man seine Webseiten mit Frames ausstatten sollte oder nicht.

Mit oder ohne Frames?

Frames ermöglichen einen recht übersichtlichen Aufbau komplexer Webdokumente und eine logische Gruppierung einzelner Angebote. Graphiken zur Navi-

gation können z.B. in einem Frame geladen werden, der sich nicht mehr verändert; und die einzelnen Dokumente in einem anderen Frame, der sozusagen als Darstellungsort dient. Wenngleich das dazu führt, daß man bestimmte Navigationselemente nicht auf jede Seite setzen muß (wodurch auch die Größe einer Seite sinkt, was die Ladezeiten verkürzt), raten wir davon ab, Frames zu verwenden; eben weil nicht alle Browser mit Frames umgehen können und weil viele Benutzer Frames ganz einfach »hassen«. Wir haben etliche Beschwerden von Besuchern unserer Homepage erhalten, die keinen framefähigen Browser verwenden oder denen Frames nicht liegen. Grundsätzlich kann man auch alles ohne Frames realisieren, und je mehr man sich auf ältere HTML-Standards beschränkt, desto größer ist die Wahrscheinlichkeit, daß die meisten Besucher Ihrer Homepage diese auch ohne Probleme betrachten können - und Ihre Seiten mögen.
Beim Besuch einer an sich interessanten Webpage stießen wir einmal auf den nicht gerade allzu freundlichen Satz des Autors, wo man denn die letzten Jahre gewesen sei, man solle sich mal *Netscape* besorgen, denn es gäbe jetzt Frames. So sollte es nicht sein. Daher empfehlen wir das Design von Webseiten ohne Frames. Wir stellen Ihnen allerdings trotzdem auch die Formatierung von Seiten mit Frames vor.

Was Sie auf unserer Homepage finden
Wir unterstützen Sie beim Design Ihrer Webseiten mit einigen Dateien, die Sie im Softwarebereich der *OS/2 Only!* zum Download auf unserer Homepage finden. Diese Dateien beinhalten folgende Hilfsmittel:

- Die hier vorgestellten Grundgerüste für Webseiten mit und ohne Frames,
- eine Icon-Sammlung zur Integration kleiner Graphiken auf Ihren Seiten,
- eine kleine Bildersammlung für Graphiken, mit denen Sie den Hintergrund Ihrer HTML-Dokumente belegen können.

Mit *Xelia 1.0* (verfügbar Anfang Herbst), erhalten Sie auch ein Vektorgraphikprogramm, mit dem Sie eigene Graphiken erstellen können. Die Navigationsgraphiken, die Sie auf unserer Homepage finden, und viele der Abbildungen in dieser Ausgabe wurden mit diesem Programm erstellt.

Anmerkungen zum Aufbau der Seiten
Bevor wir Ihnen den grundsätzlichen Aufbau eines HTML-Dokumentes zeigen, möchten wir Ihnen noch einige Hinweise geben, die Sie bei der späteren Arbeit beachten sollten:

1. Gehen Sie sparsam mit Graphiken um. Vermeiden Sie große Abbildungen hoher Farbtiefe und Auflösung. Wenn möglich, verwenden Sie am besten Graphiken mit 256 Graustufen oder 256 Farben und Auflösungen bis höchstens 92 dpi (optimal 72 dpi).
2. Bauen Sie Ihre Seiten einfach, logisch und übersichtlich auf.
3. Entwerfen Sie eine auf *jeder Seite* sofort ins Auge springende Leiste mit

Grundstruktur einer HTML-Seite

```
<HTML>

<HEAD>

[...]

</HEAD>

<BODY>

[...]

</BODY>

</HTML>
```

kleinen Graphiken oder Textlinks, die dem Besucher der Webseite eine einfache Navigation durch die wichtigsten Punkte Ihres Angebotes ermöglicht. Der beste Platz hierfür ist der Anfang der Seite oder eine Tabellenspalte am linken Seitenrand.

4. Setzen Sie die wichtigsten Links zusätzlich an das Ende einer jeden Seite, damit Besucher der Homepage auch vom Seitenende aus Ihre Webseiten weiter durchforschen können, ohne sich wieder an den Seitenanfang scrollen zu müssen.

5. Fügen Sie auf jeder Seite einen Link auf Ihre eMail-Adresse hinzu, damit man Sie bei Fragen und Kritik mit einem Klick erreichen kann.

6. Vermerken Sie, am besten am Anfang einer jeden Seite, das Datum, an dem das Dokument zuletzt aktualisiert wurde.

7. Benutzen Sie für normalen Fließtext die Standardfontgröße.

8. Verwenden Sie als Hintergrund hellgrau, weiß oder Hintergrundbilder in diesen Farbtönen. Benutzen Sie für Schrift Schwarz- oder Grautöne. Links belassen Sie beim Standard (Noch nicht besuchte Links blau, bereits besuchte, violett).

9. Vermeiden Sie, kräftige Farben auf dunkle Hintergründe zu setzen (z.B. Rot auf Schwarz, Gelb auf Blau).

Grundstruktur eines HTML-Dokuments

Jedes HTML-Dokuement hat eine festgelegte Grundsturktur. Im Listing *Grundstruktur einer HTML-Seite* sehen Sie die Tags eines Dokumentes, die unbedingt vorhanden sein *müssen*. Untersuchen wir nun diese Grundstruktur.

In HTML werden Tags mit den Zeichen < > eingeschlossen. Ferner treten Tags immer paarweise auf, wie <HTML> ... </HTML>. Der Schrägstrich / bedeutet, daß die Anweisung, die das Tag zuvor einleitete, aufgeboben ist. Mit dem Tag <B> definiert man z.B. fettgedruckten Text. Der Text, der <B>**innerhalb dieses Tags steht**</B> wird dann fettgedruckt und der Text nach dem Tag </B> wird wieder normal dargestellt.

Das Grunddokument besteht aus:

- dem Tag <HTML>, welches das HTML-Dokument begrenzt. Alle Anweisungen nach dessen Aufhebung durch </HTML> gehören nicht mehr zum Dokument.
- dem Tag <HEAD>, das einen Bereich einleitet, in dem Informatio-

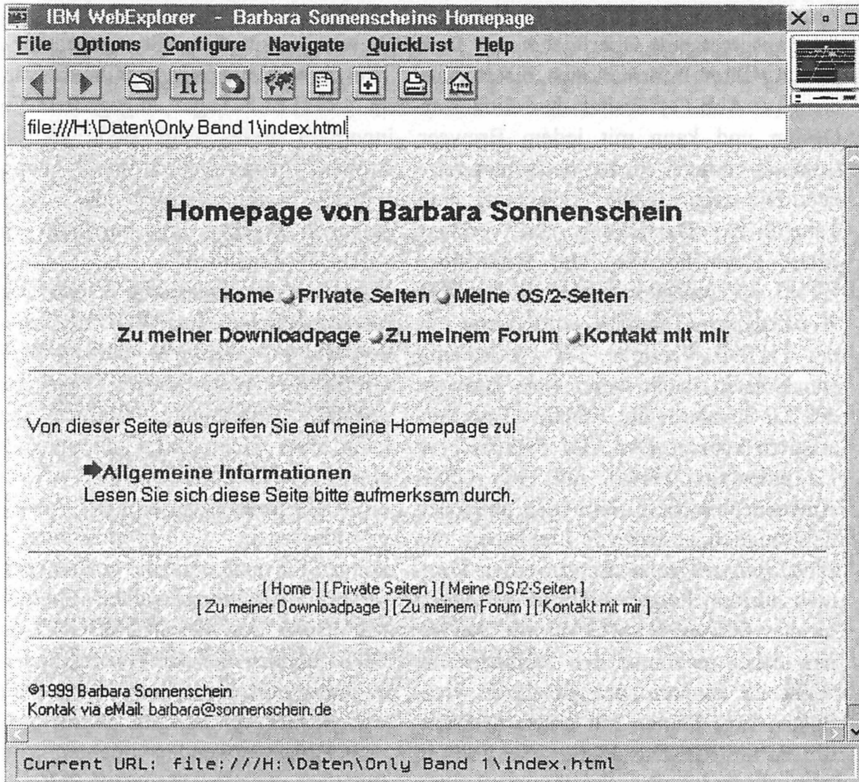


Abb. 1: Die einfachste Art, eine Homepage zu designen. Die Seiten sind zwar schlicht, können aber mit wirklich jedem Browser betrachtet werden

nen zum Dokument stehen, wie z.B. der Titel der Seite.

- Dem Tag <BODY>, das den eigentlichen Dokumentteil einleitet. Text, Bilder und Links innerhalb des <BODY> ... </BODY>-Tags gehören zu dem, was der Browser darstellt.

Das ist eigentlich schon das Wichtigste. Nun müssen wir den »Körper« des Dokumentes innerhalb des <BODY>-Tags schreiben. Damit dabei gleich etwas Sehenswertes herauskommt, haben wir drei Grundgerüste vorbereitet, die Ihnen als Vorlagen für Ihre eigenen Webdokumente dienen sollen. Wir gehen diese drei Grundgerüste nun zusammen durch, so daß es Ihnen anschließend mit Hilfe der Sprachreferenz dieses Workshops leichtfallen müßte, selbst umfangreiche Webprojekte in die Wirklichkeit umzusetzen.

Grundgerüst für Webseiten ohne Frames
In diesem Abschnitt stellen wir Ihnen zwei verschiedene Typen von Webseiten

vor, die ohne Frames auskommen und zwei ansprechende Oberflächen zur Darstellung Ihrer Informationen bieten. Die erste (s. Abb. 1) besitzt das einfachste Design und kann mit jedem Browser (und sei er noch so alt) betrachtet werden. Die zweite (Abb. 2) benutzt eine Tabelle zur Darstellung und verlangt damit einen Browser, der mindestens HTML 3.0 unterstützt, wie z.B. der *IBM WebExplorer* (Tabellen wurden erst mit der HTML-Version 3.0 vollständig standadisiert, auch wenn viele Browser des 2.0-Standards die Tabellen-Tags verarbeiten können. Der *IBM WebExplorer* 1.2 unterstützt HTML 3.0). Für OS/2-Anwender ist diese Seite also auf jeden Fall tauglich, so daß von Liebhabern des *WebExplorers* keine Beschwerden kommen dürften. Folgen wir nun der Arbeit unserer fiktiven HTML-Autorin. Sie lernen dabei nicht nur den Aufbau einer Webseite, sondern auch gleich die wichtigsten Tags kennen, mit denen man ständig zu tun hat. Das Listing der Seite in Abb. 1 finden Sie im Kasten *Listing Homepage Typ 1*.

Das Dokument wird, wie wir schon wissen, mit <HTML> eingeleitet. Im »Kopf« des Dokumentes hat unsere Webdesignerin Frau Sonnenschein den Titel ihrer Homepage mit ihrem Namen im <TITLE>-Tags angegeben. Was innerhalb dieses Tags steht, stellt das Browser für gewöhnlich in der Titelzeile seines Programmfensters dar. Weitere Angaben machen wir innerhalb des <HEAD>-Tags nicht.

Dann wird der eigentliche Teil des Dokumentes definiert. Im <BODY>-Tag finden wir ein Argument namens BACKGROUND. Viele andere Tags haben ebenfalls Argumente. Einigen davon kann man einen Wert übergeben. In einem solchen Falle wird dem Argument ein = angeschlossen, dem der Wert, eingeschlossen in " ... " folgt. Zahlenwerte können Sie auch ohne die beiden hochgestellten Anführungszeichen übergeben. Mit Hilfe des BACKGROUND-Arguments können Sie den Hintergrund Ihrer Webseite mit einer Graphik hinterlegen, wie es hier der Fall ist. Wenn Sie nur eine Farbe verwenden möchten, können Sie diese mit dem Argument BGCOLOR auswählen. Dazu übergeben Sie dem Argument BGCOLOR einen RGB-Farbwert in hexadezimaler Form. Eine

Listing Homepage Typ 1

```
<HTML>

<HEAD>
<TITLE>Barbara Sonnenscheins Homepage</TITLE>
</HEAD>

<BODY BACKGROUND="back.jpg">

<CENTER>
<H2>Homepage von Barbara Sonnenschein</H2>
```

Listing Homepage Typ 1 (Fortsetzung)

```
<FONT SIZE=4><B>
<HR>
<A HREF="index.htm">Home</A>
<IMG src="whitebal.gif" align=middle>
<A HREF="x1.htm">Private Seiten</A>
<IMG src="whitebal.gif" align=middle>
<A HREF="x2.htm">Meine OS/2-Seiten</A>
<P>
<A HREF="x3">Zu meiner Downloadpage</A>
<IMG src="whitebal.gif" align=middle>
<A HREF="x4.htm">Zu meinem Forum</A>
<IMG src="whitebal.gif" align=middle>
<A HREF="x5.htm">Kontakt mit mir</A></TR>
<HR>
</FONT></B></CENTER>

<P>Von dieser Seite aus greifen Sie auf meine Homepage zu!
<P>
<MENU>
     <IMG SRC="arrow.gif" WIDTH=13 HEIGHT=13 BORDER=0>
    <A HREF="x.htm"><B>Allgemeine Informationen</B></A>
    <BR>Lesen Sie sich diese Seite bitte aufmerksam durch.
</MENU>

<P>
<HR>
<CENTER>
<FONT SIZE=2>
<A HREF="index.htm">[ Home ]</A>
<A HREF="x1.htm">[ Private Seiten ]</A>
<A HREF="x2.htm">[ Meine OS/2-Seiten ]</A>
<BR><A HREF="x3">[ Zu meiner Downloadpage ]</A>
<A HREF="x4.htm">[ Zu meinem Forum ]</A>
<A HREF="x5.htm">[ Kontakt mit mir ]</A>
</CENTER>
<HR>
<P>&copy;1999 Barbara Sonnenschein
<BR>Kontak via eMail: <A HREF="mailto:" NAME="barbara @sonnenschein.de">
barbara@sonnenschein.de</A>
</FONT>

</BODY>

</HTML>
```

Tabelle mit den wichtigsten Farbwerten finden Sie in der Sprachreferenz.

Anschließend wird der Anfang der Seite geschrieben. Dieser »Seitenkopf« soll auf jeder Webseite stehen und ein einfaches Navigieren durch Barbara Sonnenscheins Homepage bieten. Daher wird der nun folgende Text mit dem Tag <CENTER> zentriert. Zunächst soll die Überschrift »Barbara Sonnenscheins Home - page« angezeigt werden. Dazu wird das Überschriften-Tag (Heading Tag) <H2> benutzt. Die <H..>-Tags (<H1> bis <H6>) dienen der Definition von Überschriften verschiedener Größe (1 ist hierbei am größten). Das Tag beginnt nach der Überschrift gleich einen neuen Absatz.

Der nächste Schritt besteht darin, die wichtigsten Plätze der Homepage als Textlinks darzustellen. Die Links sollen in etwas deutlicher erscheinen, also setzt Barbara die Schritgröße mit dem Tag <FONT> und dem Argument SIZE=4 eine Stufe höher (FONT=3 ist der Standardwert) und fügt das Tag <B> hinzu, um den nun folgenden Text größer und fett wiederzugeben (<B> steht für Bold).

Die Links, die Sie in der Abb. 1 sehen, sollen von der Überschrift getrennt sein. Also soll eine waagerechte, dünne Linie als Trennung eingefügt werden, was mit dem Tag <HR> erreicht wird. Dann entsteht eine waagerechte Linie über die Länge der gesamten Seite. Soll die Linie kürzer sein, können Sie sie mit dem Argument WIDTH verkleinern, dem Sie die neue Länge am besten in % übergeben, z.B. WIDTH="50%", um die Linie nur über die Hälfte der Seite reichen zu lassen.

Nun werden die sechs Links gesetzt. Dazu dient das Tag <A>, mit dem man Links aller Art definiert. Hier soll auf andere Seiten der Homepage verwiesen werden. Das ist ganz einfach: Als Argument dient hierzu HREF, dem der Name der Seite folgt, z.B. *index.htm*. Damit weiß der Browser, welches Dokument er laden und anzeigen muß, wenn der Link angeklickt wird. Genau genommen wird hinter HREF eine **URL (Uniform Resource Locator)** angegeben, also eine komplette Webadresse. Sofern man auf Dokumente oder Dateien auf dem eigenen Server verweisen will, genügt natürlich der Dateiname, ggfs. mit Pfadangabe, z.B. wären:

<A HREF="index.htm">
<A HREF="bilder.zip">
<A HREF="/page/babasun/index.htm">

für auf dem eigenen Server enthaltene Dokumente oder:

<A HREF="http://www.cefischer.de/deutsch/xelia/xelia.htm">

<A HREF="ftp://ftp.firma.de/treiber/drucker/modelle.zip">

für Dokumente oder Dateien auf anderen Servern mögliche Werte für das HREF-Argument. Wenn der Browser keinen Viewer für eine Datei definiert hat und deren Inhalt nicht anzeigen kann, bietet er an, die Datei auf den Rechner zu laden.

Alles, was das <A> Tag bis zu dessen Aufhebung mit </A> einschließt, wird als sogenannter **Hyperlink** angezeigt; man kann also darauf klicken und auf

eine andere Seite gelangen. Das kann ein Text sein, wie in diesem Fall, oder auch eine Graphik. Im folgenden werden sechs Links definiert: *Home*, mit dem man auf die Startseite *index.htm* von jeder anderen Seite aus zurückspringen kann, und fünf weitere Seiten, welche durch die HTML-Dateien *x1.htm* bis *x5.htm* defi - niert werden. Zwischen jedem Link soll eine kleine graue Kugel angezeigt werden. Das sieht gut aus und sorgt für eine deutliche Trennung. Die Kugel ist in der Bilddatei *whitebal.gif* gespeichert. Um sie anzuzeigen, dient das Tag <IMG>, dem man wie bei HREF im Argument SRC den Namen der Bilddatei übergibt, ggfs. mit Pfadbezeichnung, z.B.:

<IMG SRC="whitebal.gif">
<IMG SRC="/pictures/whitebal.gif">

In unserem Fall liegt die Bilddatei im gleichen Verzeichnis wie das HTML-Dokument *index.htm*, so daß wir wieder nur den Dateinamen anzugeben brauchen. Im <IMG>-Tag wird hier noch das Argument ALIGN zur Ausrichtung der Graphik im Text angegeben. Mit dem Wert MIDDLE wird die Graphik in der Zeilenmitte angezeigt, was für das kleine Bällchen ptimal ist. Eine Auflistung aller wichtigsten <IMG>-Argumente finden Sie in der Sprachreferenz.
So werden drei Links, getrennt durch das grauweiße Bällchen nebeneinander gesetzt. Die nächsten drei sollen aber mit einer Zeile Abstand unter den ersten stehen. Dazu wird ein neuer Absatz mit dem Tag <P> begonnen. Dadurch wird die Zeile umgebrochen und eine Leerzeile eingefügt. Möchte man nur einen Zeilenumbruch erreichen, verwendet man das Tag
, das ebenfalls noch in diesem Dokument vorkommt.
Schließlich wird der »Seitenkopf zur Navigation« mit einem abermaligen <HR> zur Erzeugung einer neuen Trennlinie abgeschlossen und die zuvor gesetzten Tags zur Textformatierung mit </FONT>, </B> und </CENTER> aufgehoben.
Nun folgt der individuelle Teil dieser Seite. Frau Sonnenscheins Seiten befinden sich noch im Aufbau, deswegen faßt sie sich kurz, beginnt mit <P> einen neuen Absatz, fügt einen kurzen Text an und bietet nach einem neuen Absatz bereits einen Auswahlpunkt mit dem <MENU>-Tag. Dies bewirkt eine Einrückung, die mit einem kleinen roten Pfeil gekennzeichnet wird. Im bereits bekannten <IMG>-Tag finden Sie drei neue Argumente: Zunächst WIDTH und HEIGHT, mit denen die Breite und Höhe der Graphik festgelegt wird, entweder in Pixeln (wie hier im Beispiel) oder in %, wobei man sich dann auf die bereits gegebene Höhe und Breite bezieht. Läßt man diese beiden Argumente fort, so wird die Graphik in ihrer Originalgröße angezeigt. Das dritte neue Argument ist BORDER, mit dem man die Stärke des Rahmens angibt, der um die Graphik gezeichnet werden soll. Ein Wert von 0 unterdrückt das Zeichnen eines Rahmens (die Werte 1 bis 6 sind erlaubt, wobei 1 der schmalste Rahmen ist). Normalerweise braucht man dieses Argument nicht, weil Graphiken standardmäßig ohne Rahmen gezeichnet werden (was aber nicht so sicher ist, es mag Browser geben, die dies nicht tun). Wichtig wird

es aber, wenn Sie eine Graphik einen Link werden lassen. Dann wird nämlich auf jeden Fall ein Rahmen gezeichnet, und das ist in den meisten Fällen häßlich. Mit BORDER=0 können Sie den Rahmen dann verschwinden lassen.
Nach dem kleinen Pfeil soll wieder ein Link folgen, der fett dargestellt werden soll. Soweit ist alles bekannt. Dem Menüpunkt soll aber noch eine Erklärung folgen, die direkt darunter in einer anderen Zeile steht. Wie bereits erwähnt erzeugt man einen Zeilenumbruch mit dem Tag
. Ihm folgt der Text, mit dem die neue Zeile beginnt. Dann hebt Barbara das <MENU>-Tag (also die Einrückung) mit </MENU> auf.
In diesen Bereich der Seite sollen einmal die eigentlich Informationen aufgenommen werden, also Graphiken, Dateien zum Download und Texte. Vorerst soll das Grundgerüst aber noch durch einen »Fußteil« komplett werden.
Darin bietet Frau Sonnenschein noch einmal die wichtigsten Links vom Seitenkopf, außerdem einen Copyrightvermerk und ihre eMail-Adresse (über die ihr Benutzer des *WebExplorers* viel Lob zumailen, weil sie keine Frames verwendet).
Der Fußteil beginnt mit einem neuen Absatz (<P>) und einer waagerechten Linie, um ihn von der übrigen Seite abzutrennen. Weil die Links zentriert dargestellt werden sollen, folgt das bekannte <CENTER>-Tag. Sie sollen am Seitenende auch nicht zuviel Platz wegnehmen, also wird die Schriftgröße mit dem Tag <FONT> wieder verändert, in diesem Falle mit SIZE=2 um einen Schritt verkleinert. Kleiner als 2 sollte die Schrift auf einer Webseite übrigens nicht sein. Innerhalb dieses Anweisungsblocks folgen dann wieder sechs Links (diesmal nur mit einem Zeilenumbruch). Da der Copyrightvermerk und die eMail-Adresse wieder linksbündig angezeigt werden sollen, wird die Zentrierung mit </CENTER> aufgehoben. Der Rest des Fußteils besteht aus bekannten Dingen. Neu ist jedoch die Zeichenkombination © Dazu muß man wissen, daß Sonderzeichen aller Art (s. Tabelle in der Sprachrefernz) mit einem & eingeleitet und einem ; abgeschlossen werden. Ein © läßt den Browser ein © darstellen. Auch Umlaute und dergl. gehören zu den Sonderzeichen. Ein ä wird zu einem ä und ein ß zu einem ß
Eine weitere Neuheit ist ein unbekannter Wert im HREF-Argument des Links und ein neues Argument namens NAME. Da wir eine eMail-Adresse angeben möchten, teilen wir dies dem Browser mit einem HREF="mailto:" mit. Bei einem Klick auf den Link kann der Browser dann das Mailto-Protokoll starten. Um zu wissen, an wen die eMail gesendet werden soll, gibt man im Argument NAME die komplette eMail-Adresse an. Das NAME-Argument bezeichnet eigentlich Sprungmarken, mit denen man an verschiedene Stellen innerhalb eines Dokumentes springen (s. Sprachreferenz) kann. In diesem Falle ist die Sprungmarke die eMail-Adresse.
Damit ist das Gerüst der Webseite fast komplett. Nur noch die zuvor gesetzte Fontgröße wird mit </FONT> aufgehoben, der »Körper« des Dokumentes mit </BODY> beendet und das gesamte Dokument mit </HTML> abgeschlossen.

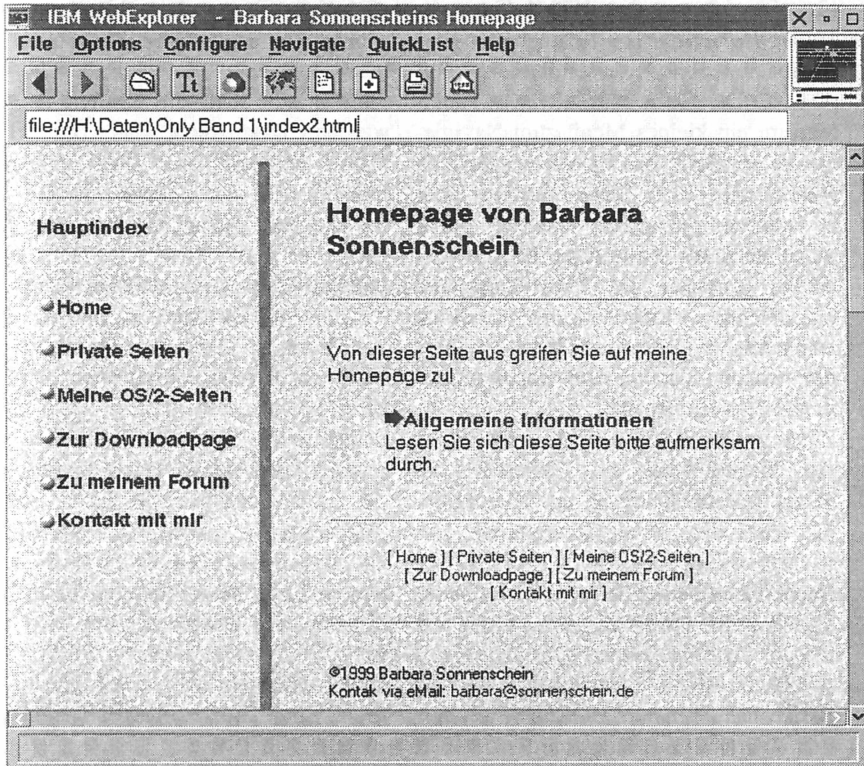


Abb. 2: Eine etwas schönere, aber immer noch einfach zu erstellende Webseite. Zwar sieht diese Seite etwas einladender aus als die erste, dafür braucht man einen Browser, der mindestens HTML 3.0 unterstützt.

Das soeben vorgestellte Dokument zeigt die einfachste Art, eine Homepage zu erstellen. Zwischen dem Kopf- und dem Fußteil der einzelnen Seiten werden die eigentlichen Daten eingefügt. Weil man die ganze Seitenbreite zur Verfügung hat, bringt man dort eine Menge unter. Außerdem besticht die Seite durch eine gewisse »schöne Schlichtheit«. Mit einigen kleinen Graphiken läßt sie sich noch beträchtlich verbessern und schnell erweitern. Zum Beispiel könnte man die Textlinks im Seitenkopf durch geeignete Graphiken ersetzen. Auf jeden Fall ist ein solches Design immer dann zu vorzuziehen, wenn man möchte, daß die Seite bestimmt mit jedem Browser, v.a. auch mit einem älteren, problemlos zu betrachten ist. Man wird bei diesen Seiten auch die geringsten Abweichungen in der Darstellung durch die verschiedenen Browser feststellen.

Der nächste Webpagetyp sieht ein wenig ansprechender aus als das erste Grundge-

rüst. Sie können den HTML-Text aus *Listing Homepage Typ 2* entnehmen. Wie das Dokument im Browser aussieht, zeigt Abb. 2. Die Seite definiert zwei Tabellen, um ein kleines Menü zum Navigieren zu erstellen, das in einem schmalen Bereich am linken Seitenrand verfügbar ist; während im rechten Seitenteil -durch einen schmalen senkrechten Balken vom ersten getrennt- die Darstellung der eigentlichen Informationen erfolgt. Betrachten wir nun den HTML-Quelltext der zweiten Version von Barbara Sonnenscheins Homepage.

Zunächst ist alles gleich: Die Einleitung des Dokumentes, die Definition des Titels und die Belegung des Hintergrundes durch eine Backgroundgraphik. Das nächste Tag <TABLE> dient der Beschreibung einer der bereits erwähnten Tabellen. Sie nimmt die sechs Links auf, die wir bereits kennengelernt haben, und den senkrechten Trennstrich. Das Tag besitzt das Argument BORDER mit dem dem Wert 0. Die Bedeutung ist hier die gleiche wie beim Tag <IMG>; mit 0 erzeugen wir eine Tabelle ohne Rahmen. Von 1 bis 6 wird sowohl um die gesamte Tabelle als auch um jede Zelle ein Rahmen gezeichnet. CELLSPACING legt den Abstand zwischen zwei Zellen in der Tabelle fest. Ein Wert von 5 bewirkt im Falle Frau Sonnenscheins Menüs, daß der Farbbalken nicht an die Schrift der Menüpunkte stößt. Das nächste Argument ALIGN kennen wir ebenfalls vom Tag <IMG>. ALIGN erlaubt auch hier eine Ausrichtung im Text, in diesem Falle der Tabelle. Der Wert LEFT bewirkt eine Plazierung der Tabelle am linken Seitenrand. Damit ist das <TABLE>-Tag abgeschlossen. Das nächste Tag heißt <TR>. Mit diesem Tag leitet man eine Tabellenzeile ein. Nun müssen nur noch die Spalten definiert werden. Dazu dient das Tag <TD>. Damit hat man alle Tags, die man braucht: Soll eine Tabelle z.B. drei Zeilen und fünf Spalten haben, so benötigte man drei mal <TR> ... </TR>, wobei innerhalb eines jeden solchen Paares fünf <TD> ... </TD>-Paare zu finden wären. Auf Barbara Sonnenscheins Homepage wird eine Tabelle mit nur einer Zeile und zwei Spalten erstellt. Die erste Spalte beinhaltet die Links; die zweite den senkrechten Trennbalken. Daher finden Sie in der Tabelle auch nur ein <TR> ... </TR>-Tagpaar und zwei <TD> ... <TD>-Paare. Die einzelnen Zeilen für die Links erzeugen wir mit <P>-Tags. Innerhalb einer Tabellenzelle können Sie den Text nämlich frei formatieren und natürlich auch Links einfügen. Nur eines können Sie nicht: Eine Tabelle in einer Tabelle definieren.

Im <TD>-Tag finden Sie wieder einige Argumente. Sie dienen dazu, den Text innerhalb des folgenden Tabellenelements zu formatieren. Die Bedeutung von ALIGN kennen Sie bereits. VALIGN jedoch ist neu. Es beschreibt die vertikale Ausrichtung des Tabellentextes. Standardmäßig wird der Text vertikal zentriert dargestellt. Wir möchten, daß der Text zum oberen Rand der Tabellenzeile -in diesem Falle der gesamten Tabelle- ausgerichtet wird. Also belegen wir VALIGN mit dem Wert TOP. Das dritte Argument NOWRAP verhindert einen Umbruch des Textes, wenn die Tabelle mal verkleinert wird.

Listing Homepage Typ 2

```
<HTML>

<HEAD>
<TITLE>Barbara Sonnenscheins Homepage</TITLE>
</HEAD>

<BODY BACKGROUND="back.jpg">

<TABLE BORDER=0 CELLSPACING=5 ALIGN=LEFT>

<TR>
<TD VALIGN=TOP ALIGN=LEFT NOWRAP>
<HR>
<B>Hauptindex</B>
<HR>
<FONT SIZE=4>
<B>
<P><IMG src="whitebal.gif" align=middle> <A HREF="index.htm">Home</A>
<P><IMG src="whitebal.gif" align=middle> <A HREF="x1.htm">Private Seiten</A>
<P><IMG src="whitebal.gif" align=middle> <A HREF="x2.htm">Meine OS/2-Seiten</A>
<P><IMG src="whitebal.gif" align=middle> <A HREF="x3.htm">Zur Downloadpage</A>
<P><IMG src="whitebal.gif" align=middle> <A HREF="x4.htm">Zu meinem Forum</A>
<P><IMG src="whitebal.gif" align=middle> <A HREF="x5.htm">Kontakt mit mir</A>
</B></FONT>
</TD>

<TD>
<IMG src="vline.gif" align=middle>
</TD>
</TR>

</TABLE>


<TABLE BORDER=0 WIDTH=90%>

<TR>
<TD VALIGN=TOP>
<H2>Homepage von Barbara Sonnenschein</H2>
<HR>
<P>Von dieser Seite aus greifen Sie auf meine Homepage zu!
<P>
```

Listing Homepage Typ 2 (Fortsetzung)

```
<MENU>
   <IMG SRC="arrow.gif" WIDTH=13 HEIGHT=13 BORDER=0>
   <A HREF="x.htm"> <B>Allgemeine Informationen</B></A>
   <BR>Lesen Sie sich diese Seite bitte aufmerksam durch.
</MENU>

<P>
<FONT SIZE=2>
<HR>
<CENTER>
<A HREF="index.htm">[ Home ]</A>
<A HREF="x2.htm">[ Private Seiten ]</A>
<A HREF="x3.htm">[ Meine OS/2-Seiten ]</A>
<BR><A HREF="x6">[ Zur Downloadpage ]</A>
<A HREF="x7.htm">[ Zu meinem Forum ]</A>
<BR><A HREF="x7.htm">[ Kontakt mit mir ]</A>
</CENTER>
<HR>
<P>&copy;1999 Barbara Sonnenschein
<BR>Kontak via eMail: <A HREF="mailto:" NAME="barbara @sonnenschein.de">
barbara@sonnenschein.de</A>
</FONT>

</TD>
</TR>
</TABLE>

</BODY>

</HTML>
```

Damit behalten die Links das angenehme Bild, das sie jetzt haben. Wenn Sie dieses Homepagegerüst jedoch für Ihre Webseiten verwenden, beachten Sie bitte, den Links keinen allzu langen Namen zu geben. Da der Browser den Text in den Tabellenelementen nicht umbrechen darf, wird der Menübereich breiter, und der Bereich zur Darstellung der eigentlichen Informationen schmaler, was natürlich nicht im Sinne des Erfinders liegt. Frau Sonnenschein hat daher auch einen ihrer Links, nämlich *Zu meiner Downloadpage* in das kürzere *Zur Downloadpage* geändert.

Nach diesen Tabellenbeschreibungen wird die Tabelle mit Text und Graphiken gefüllt. Zuerst mit zwei horizontalen Linien, die einen mit <B> ... </B> fettgedruckten Hauptindex aufnehmen sollen. Daran anschließend wird wieder mit <FONT SIZE=4> die Schrift der Links vergrößert und das <B>-Tag gesetzt, um die Links fettgedruckt darzustellen. Dann

werden die Links wie gehabt angegeben, wobei Barbara nun vor jeden eines der grauweißen Bällchen gesetzt hat. Schließlich wird die Tabellenspalte mit einem </TD> abgeschlossen.
Nachdem Schriftstil und Fontgröße mit </B> und </FONT> ebenfalls aufgehoben wurden, wird mit <TD> eine neue Tabellenspalte definiert, welche den vertikalen Trennbalken mit einem IMG-Tag aufnimmt. Mehr soll dort nicht dargestellt werden, also wird mit </TD> zuerst die neue Spalte und dann mit </TR> die Zeile beendet, um letztlich mit </TABLE> die Beschreibung der ersten Tabelle abzuschließen.
Den eigentlichen Inhalt der Webseite nimmt die zweite Tabelle auf. Sie besteht nur aus einer Zeile und einer Spalte. In dem der ersten Tabelle folgenden <TABLE>-Tag finden Sie ein neues Argument namens WIDTH, mit dem die Breite der Tabelle festgelegt wird, entweder in Pixeln oder in einer auf die Seitenbreite bezogenen relativen Angabe in %. Barbara gibt hier 90 % an, um die Tabelle fast den gesamten noch sichtbaren Bereich der Seite einnehmen zu lassen. Würde eine Angabe von WIDTH unterbleiben, legt der Browser die Tabellenbreite fest, und in diesem Fall würde sie zu schmal.
Dann folgt die bereits bekannte Definition der Tabellenzeile und der Tabellenspalte. Das Argument VALIGN im Tag <TD> bewirkt hier wieder, den Tabelleninhalt am oberen Rand der Tabelle darzustellen. Der HTML-Text, der dann folgt, ist bereits bekannt. Im wesentlichen wurde durch die Einführung der beiden Tabellen der Kopfteil der Seite in das Menü am linken Seitenrand verlagert, Der Inhalt, wie er schon in der ersten Version bestand, ist derselbe geblieben, ebenso wie der Fußteil, der nur um ein
-Tag bereichert wurde.
In diesem Bereich nehmen Sie sämtliche Informationen auf, wenn Sie Ihre Homepage mit diesem Design versehen wollen. Den Fußteil integrieren Sie wie beim ersten Grundgerüst in jeder Ihrer Seiten, damit der Besucher die Möglichkeit hat, vom unteren Seitenrand zu den anderen Stellen Ihrer Homepage zu springen.
Das Homepagegerüst wird abgeschlossen durch die </TD>, </TR> und </TABLE> Tags, um die Definition der Tabelle zu beenden; und wie gewohnt mit </BODY> und </HTML>, um dem Dokument einen Abschluß zu geben.
Dieser Homepagetyp bietet vielfältige Veränderungsmöglichkeiten. Sie können z.B. den Hauptindex ganz problemlos erweitern, indem Sie den ein oder anderen Link hinzufügen; vielleicht auch hier als kleine Graphiken, was die Webseite noch ein wenig interessanter macht. Der große Vorteil besteht darin, daß jede Seite ihren eigenen Index bekommen kann. So wäre es möglich, den Hauptindex auf jede Seite zu nehmen und Nebenindizes für die einzelnen Bereiche Ihrer Homepage, wenn man sich in einem dieser Bereiche befindet. Das erleichtert ein Navigieren ungemein.
Ein weiterer Vorteil dieses Webseitentypus gegenüber dem ersten besteht darin, daß die Textzeilen nicht zu lang werden, da der Darstellungsbereich durch das Menü etwas schmaler ist. Das trägt zu einem angenehmen Lesen bei, weil das Auge eine Zeile meist vor ihrem Ende

umbrechen »möchte«. Das erklärt auch, warum Zeitungen und Magazine, wie auch die *OS/2 Only!*, im Spaltensatz gedruckt sind. Das zweite Webseitengerüst ermöglicht ein optimales Lesen.

Was bei beiden Seiten zu beachten ist
Wenn Sie die beiden Grundgerüste zum Design einer eigenen Homepage einsetzen möchten, beachten Sie bitte folgendes:

- Die erste Seite (die Startseite) müssen Sie richtig benennen, damit der Server sie finden kann. Sprechen Sie mit Ihrem Provider ab, wie diese Seite heißen muß (meistens *index.html* oder *home.html*).

- Verändern Sie eines der vorgegebenen Muster nach Ihrem Geschmack und verwenden Sie es als Schablone für weitere Webseiten. Kopieren Sie diese Datei dazu in ein für Ihr Webdesign angelegtes Verzeichnis, in dem Sie auch alle anderen Dateien ablegen. Ihre Homepage betreffende Verzeichnisstrukturen legen Sie in diesem Verzeichnis an. Vermeiden Sie es, die einzelnen Dateien über die Platte zu verstreuen.

- Legen Sie, bevor Sie an das Gestalten gehen, eine übersichtliche Struktur Ihrer Webseiten fest. Sofern Ihr Provider Ihnen diese Möglichkeit bietet, legen Sie Unterverzeichnisse auf Ihrem Server oder in Ihrem Verzeichnis an, um zusammengehörige Dateien zu gruppieren. Gerade bei umfangreichen Projekten hilft das, den Überblick zu behalten. Spätere Veränderungen der Sturktur wirken sich auf alle Links und u.U. auf viele Dokumente aus. Solche Änderungen sind sehr zeitintensiv.

- Belassen Sie es dabei, alle Tags groß zu schreiben, das hebt sie vom übrigen Text ab und erleichtert die Fehlersuche.

- Schreiben Sie am besten alle Dateinamen klein. Unix-Server z.B. beachten die Groß- und Kleinschreibung, und die ist bei vielen Dokumenten schwierig einhaltbar. Also: Alles klein schreiben und vor dem Aufspielen auf den Server darauf achten, daß dann auch alle Dateinamen komplett kleingeschrieben sind.

Wir verwendeten Barbara Sonnenscheins zweites Grundgerüst für unsere Webseiten und gestalteten es natürlich noch entsprechend aus. Letztlich ist es natürlich Geschmacksache, zu welchem Design man eher neigt. Im folgenden Abschnitt befassen wir uns mit einem Grundgerüst für eine Homepage, die Frames verwendet.

Grundgerüst für Webseiten mit Frames
Mit dem dritten unserer Grundgerüste verlassen wir den Anwendungsbereich des IBMWebExplorers, da wir nun Tags in unseren Webdokumenten verwenden, die erst ab HTML 3.2 standadiesiert wurden. Dazu gehören die Frames. Sie ermöglichen es, ein Dokument in mehrere voneinander unabhängige Bereiche zu unterteilen. Für unser zweites Grundgerüst würde dies bedeuten, daß das Menü in einem Teil der Seite immer

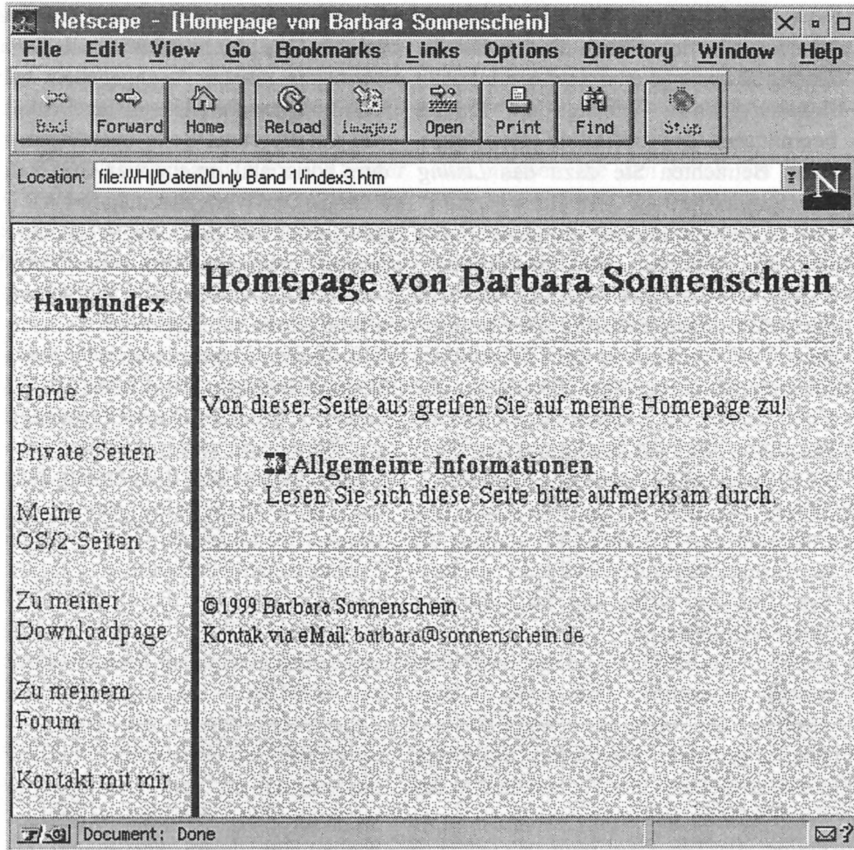


Abb. 3: Barbara Sonnenscheins Homepage mit Frames. Frames ermöglichen eine saubere Darstellung unterschiedlicher HTML-Dokumente in einem Fenster. Allerdings ist ein Browser erforderlich, der HTML ab 3.2 beherrscht.

sichtbar ist; und der Inhalt der Webpage in einem anderen, vom Menü unabhängingen Teil. Damit würde das Navigationsmenü nicht immer verschwinden, wenn man die Seite nach unten scrollt. Frames stellen Rahmen im Darstellungsbereich des Browsers dar. Innerhalb dieser Rahmen werden verschiedene HTML-Dokumente angezeigt. Der Aufbau einer Webseite mit Frames erfordert daher etwas mehr Überlegung. Wie Frau Sonnenscheins Seite mit Frames aussieht, zeigt Abb. 3. Um eine solche Seite zu erstellen, benötigt man verschiedene HTML-Dokumente. Eines heißt index.html o.ä. und ist die Startseite. Diese Seite definiert die einzelnen Frames. Die anderen HTML-Dokumente definieren den Inhalt der einzelnen Frames. In unserem Beispiel brauchen wir 3

HTML-Dokumente. Den HTML-Quelltext für alle drei Seiten finden Sie im *Listing Homepage Typ 3 a)* bis *3c)*. Die Erstellung einer Webpage mit Frames beginnt mit der Definition der Hauptseite. Betrachten Sie dazu das *Listing Homepage Typ 3 a)*.

Das Dokument beginnt wie gewohnt mit dem Tag <HTML>, und im <HEAD>-Tag wird wieder mit <TITLE> der Titel der Seite angegeben. Der hier angegebene Titel erscheint in der Titelzeile des Programmfensters des Browsers, die Titel, die Sie auf anderen Seiten definieren, nicht. Nach der Festlegung des Titels folgt etwas Neues: das Tag FRAMESET. Dieses Tag steht für das <BODY>-Tag. Innerhalb von FRAMESET werden die einzelnen Frames der Webseite beschrieben. Das Tag selbst legt fest, wie eine Seite aufzuteilen ist.

Im Tag FRAMESET sind mehrere Argumente angegeben. Das erste, COLS, bestimmt die Breite des ersten Frames in Pixeln, gemessen vom linken Seitenrand aus zum rechten. Wir übergeben ihm den Wert "120,*" und erzeugen damit einen Frame, der 120 Pixel breit ist. Der Rest des Fensters wird vom zweiten Frame vollständig ausgefüllt, da wir als zweiten Wert von COLS * angegeben haben. Möchte man die Seite nicht vertikal, so wie auf Barbaras Seite unterteilen, sondern horizontal, ersetzt man COLS durch ROWS. Der Wert, den man ROWS übergibt, entspricht dann der Breite des Frames in Pixeln, gemessen vom oberen Seitenrand zum unteren. Die Werteübergabe erfolgt genau wie bei COLS.

Mit diesem Argument wurden nun zwei Frames erzeugt. Mit FRAMEBORDER und dem Wert YES wird festgelegt, das die Frames mit einem Rahmen umgeben werden sollen. Wie breit er sein soll, legt das Tag BORDER fest, das wir bereits kennen. Im folgenden werden die beiden Frames genauer definiert.

Das geschieht über die <FRAME>-Tags, die <FRAMESET> ... </FRAMESET> einschließt. Für jeden Frame müssen Sie ein <FRAME>-Tag angeben. Zwei der Argumente kennen Sie bereits: SRC bezeichnet den Namen des HTML-Dokumentes, das in dem Frame dargestellt werden soll; NAME vergibt einen Namen für dieses Dokument. Diesem

Listing Homepage Typ 3 a)

```
<HTML>

<HEAD>
<TITLE>Homepage von Barbara Sonnenschein</TITLE>
</HEAD>

<FRAMESET COLS="120,*" FRAMEBORDER="YES" BRODER="2">
<FRAME SRC="navigate.htm" NAME="nav" SCROLLING="no" NORESIZE>
<FRAME SRC="main.htm" NAME="main"  SCROLLING="auto" NORESIZE>
</FRAMESET>

</HTML>
```

Argument *müssen* Sie einen Wert übergeben, sonst funktionieren spätere Links nicht richtig. Die anderen Argumente dienen dazu, das Verhalten des Frames festzlegen.

SCROLLING kann mit YES, NO oder AUTO belegt werden. Das Argument bestimmt, wann Scrollbars angezeigt werden sollen, um verdeckte Bereiche eines Frames wieder sichtbar machen zu können. YES zeigt die Scrollbars am Rand des Frames immer an; NO schaltet sie ganz ab. Frau Sonnenschein wählt AUTO, damit der Browser die Scrollbars bei Bedarf anzeigt bzw. wieder verdeckt. Das Tag NORESIZE bewirkt, daß der Besucher der Homepage die Größe des Rahmens mit der Maus nicht verändern kann. Damit sind die beiden Frames definiert. </FRAMESET> schließt die Definition, </HTML> das Dokument ab.

Nun müssen Sie den Inhalt der beiden Frames definieren. Im ersten Listing wurden zwei HTML-Dokumente benannt: *navigate.htm* und *main.htm*. Das erste Dokument enthält die sechs Links, die wir von den anderen Dokumenttypen her kennen. *Listing Homepage Typ 3 b)* zeigt die Links noch einmal im neuen Dokument. Die einzelnen Tags dürften mittlerweise bekannt sein. Neu ist lediglich ein Argument namens TARGET im <A>

Listing Homepage Typ 3 b)

```
<HTML>

<HEAD>
<TITLE>Barbara Sonnenscheins Homepage</TITLE>
</HEAD>

<BODY BACKGROUND="back.jpg">

<BR><HR>
<CENTER>
<B>Hauptindex</B>
</CENTER>
<HR>

<P><A HREF="main.htm" TARGET="main" >Home</A>
<P><A HREF="x1.htm" TARGET="main">Private Seiten</A>
<P><A HREF="x2.htm" TARGET="main">Meine OS/2-Seiten</A>
<P><A HREF="x3.htm" TARGET="main">Zu meiner Downloadpage</A>
<P><A HREF="x4.htm" TARGET="main">Zu meinem Forum</A>
<P><A HREF="x5.htm" TARGET="main">Kontakt mit mir</A>


</BODY>


</HTML>
```

Tag. Das TARGET-Argument gibt den Namen des HTML-Dokumentes an, das zuvor innerhalb des <FRAME>-Tags gewählt wurde. Dieses Dokument dient als Ziel bei Aktivierung des Links, d.h. es wird gegen das durch Anklicken des Links gewählte Dokument ersetzt. Alle Links, die auf den Seiten Frau Sonnenscheins Homepage enthalten sind, müssen das Argument TARGET beinhalten und als Target *"main"* angeben. Der Frame *"nav"* ist nie Target. Das Dokument in diesem Frame soll schließlich immer sichtbar sein, damit man, egal, wo man sich auf der Homepage befindet, immer den Hauptindex zur Verfügung hat.

Das Dokument *main.htm* beinhaltet im wesentlichen genau das gleiche wie die beiden Hauptseiten der anderen Dokumenttypen. Lediglich der Fußteil der Seite ist bis auf den Copyrightvermerk und Barbaras eMail-Adresse gestrichen worden. Die Wiederholung der wichtig-

Listing Homepage Typ 3 c)

```
<HTML>

<HEAD>
<TITLE>Barbara Sonnenscheins Homepage</TITLE>
</HEAD>

<BODY BACKGROUND="back.jpg">

<BR><H2>Homepage von Barbara Sonnenschein</H2>
<HR>
<P>Von dieser Seite aus greifen Sie auf meine Homepage zu!
<P>
<MENU>
    <IMG SRC="arrow.gif" WIDTH=13 HEIGHT=13 BORDER=0>
    <A HREF="x.htm" TARGET="main"> <B>Allgemeine Informationen</B></A>
   <BR>Lesen Sie sich diese Seite bitte aufmerksam durch.
</MENU>

<P>
<HR>
<FONT SIZE=2>
<P>&copy;1999 Barbara Sonnenschein
<BR>Kontak via eMail: <A HREF="mailto:" NAME="barbara @sonnenschein.de">
barbara@sonnenschein.de</A>
</FONT>


</BODY>

</HTML>
```

sten Links am Seitenende ist nicht länger nötig, da der linke Frame das Menü zur Navigation immer anzeigt, egal auf welcher Seite man sich gerade befindet. Zu beachten ist lediglich das TARGET-Argument im Link, dessen Bedeutung bereits besprochen wurde.

Weitere Hinweise zu Frames

Manchmal kommt man mit einem Frame nicht aus. Vielleicht benötigt man noch einen Index, um die Navigation durch die Seiten zu optimieren. Auf Frau Sonnenscheins Homepage böte sich an, den großen Bereich zur Darstellung, der durch das Dokument *main.htm* beschrieben wird, noch einmal horizontal zu unterteilen. Dazu gehen Sie folgt vor:

❶ Sie ersetzen den Inhalt des Dokumentes *main.htm* durch eine FRAMESET-Definition wie sie bereits beschrieben wurde.

❷ Definieren Sie im <FRAMESET>-Tag zwei Frames mit dem Argument ROWS, z.B. *ROWS="100,*"*.

❸ Weisen Sie den beiden Frames die HTML-Dokumente *main1.htm* und *main2.htm* zu, und geben Sie ihnen die Namen *main1* und *main2*.

❹ Fügen Sie den alten Inhalt von *main.htm* in *main2.htm* ein. Schreiben Sie ein neues HTML-Dokument, das für die Navigation geeignet ist, und speichern Sie es unter *main1.htm*.

❺ Für alle Seiten, die neben der Startseite den neuen Frame als Index verwenden sollen, verfahren Sie genau so wie zuvor. Damit entstehen für jede alte Seite 3 neue Seiten: Eine, welche die Framedefinitionen enthält; und zwei, welche die Frames ausfüllen.

❻ Um von irgend einer Seite aus nur den Inhalt des Darstellungsbereiches (*main2*) durch eine Seite *xxx2.htm* zu verändern, die Seite *main1* aber beizubehalten, geben Sie als *Href* im Link *xxx2.htm* und als *Target main2* ein.

❼ Um sowohl *main2* als auch *main1* durch zwei neue HTML-Dokumente *xxx1.htm* und *xxx2.htm* zu ersetzen, geben Sie als *Href* im Link *xxxx.htm* und als *Target main* an.

Wie sie sehen, wird der Aufwand der Seitenerstellung um so größer, je mehr Frames Sie erstellen. Dafür kann man seine Homepage sehr gut strukturieren. Jedoch darf trotz allem nicht außer acht gelassen werden, daß Frames von vielen nicht gemocht werden und von manchen Browsern nicht dargestellt werden können, so daß es besser ist, zu einer der Homepagevorlagen zurückzugreifen, die ohne Frames auskommen.

Im nächsten Teil unseres Workshops erhalten Sie die angekündigte Sprachreferenz, die zusammen mit den drei Beispielen ein problemloses Zusammenstellen Ihrer Webseiten ermöglichen dürfte. Zusätzlich haben wir für Sie einige HTML-Editoren untersucht und geben Ihnen Tips, welche dieser Programme nützlich für der Arbeit sind. ❑

ASTRAC Visualizer

Auf den folgenden Seiten stellt Ihnen die Firma *CONCEPT System & Software Development GmbH* den *ASTRAC Visualizer* vor, ein interessantes Produkt für *DB2*. Wir werden uns das Programm in einem späteren Test auch noch näher ansehen, lassen nun aber erst einmal *CONCEPT* mit der Vorstellung des Produktes zu Wort kommen.

Visualizer für DB2-Daten unter OS/2

Als Anwender von OS/2 werden Sie die neue Produktfamilie des *ASTRAC Visualizer* lieben. *Visualizer* für OS/2 sorgt für eine verblüffend einfache Analyse der *DB2*-Daten Ihres Unternehmens. Mit *Visualizer* können Sie nun auch Adhoc-Abfragen durchführen und danach mit Hilfe der integrierten Berichts- und Grafikfunktionen kombinieren, konsolidieren und analysieren. Entdecken Sie, wie die leistungsfähigen Business-Intelligenz-funktionen des *Visualizer* Ihnen die »Geheimnisse« Ihrer Daten enthüllen und einen Weg zum besseren Verständnis Ihrer DB2-Daten eröffnen. *Visualizers* leistungsstarke Analysefähigkeiten -die normalerweise nur in sehr teueren High-End-Produkten des Data Mining zur Verfügung stehen- sind nun zu einem Budgetpreis von *ASTRAC* erhältlich.

Die Entwicklung von *Visualizer* begann 1986 durch IBM UK in Warwick, England - der Heimat des Application System (AS). *Visualizer* wurde ursprünglich »Personal AS (PAS)« genannt und für OS/2 v1.0 herausgebracht, lange vor dem *Presentation Manager* oder der *Workplace Shell*. Zwei weitere Versionen erlebten die Namensänderung in das preisgekrönte Produkt *Visualizer* im Jahr 1994.

Visualizer war wegweisend im OS/2- und DB2 Support mit einer der ersten, und wahrscheinlich heute besten, Workplace Shell Integration. *ASTRAC*, nun der Besitzer von *Visualizer* und AS, nahm neue Erweiterungen vor. *Visualizer v1.3* von *ASTRAC*, das aktuelle Release, verlängert das Leben aller OS/2 und *DB2* Anwendungen in bedeutendem Maße mit großartigen Funktionsverbesserungen und bedeutenden Performancesteigerungen.

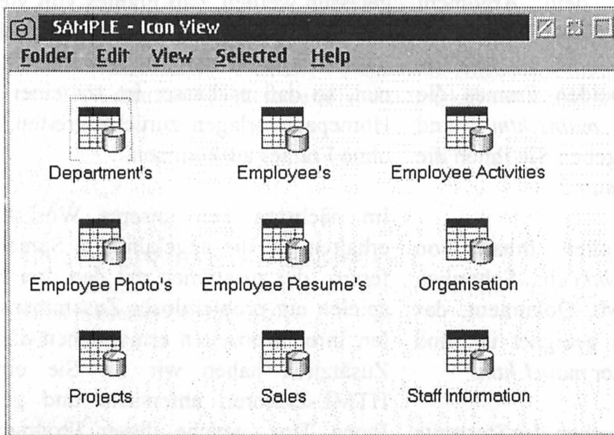


Abb. 1: DB2-Datenbanken mit Visualizer auf einen Blick

Zur Programmfamilie des *ASTRAC Visualizer v1.3* OS/2 gehören:

- ASTRAC Visualizer Query für OS/2 (VQ2v1.3)
- ASTRAC Visualizer Procedures für OS/2 (VP2v1.3)
- ASTRAC Visualizer Application Development für OS/2 (VAD2v1.3)
- ASTRAC Visualizer Query für OS/2 (VQ2v1.3)

Visualizer Query bietet anspruchsvollen Anwendern die ausgereifte Analyse, die sie benötigen. VQ2 ist so benutzerfreundlich, daß jeder OS/2 »Endbenutzer« ganz einfach Antworten auf seine Geschäftsanforderungen finden kann. Für *Visualizer* müssen Sie kein SQL-Experte sein! Aber auch SQL-Experten, die das Schreiben ihrer eigenen SQL-Anweisungen bevorzugen, wird *Visualizer* gefallen. Der Komplexität von *Visualizer* SQL-Anweisungsobjekten sind keine Grenzen gesetzt und durch das im Programm enthaltene SQL-Datenbank-Ordnerobjekt, werden Datenbankordner, in denen Daten mit Zugriffsberechtigungen sind, in Symbolform dargestellt (Abb.1). Dies macht das Auffinden und Anzeigen Ihrer SQL-Tabellen ganz einfach. Mit der entsprechenden Berechtigung können Sie ebenfalls Datenbanken erstellen und löschen.

Entdecken Sie, wie die ungezählten Formatierfunktionen Ihnen die Ansicht von Daten auf jede von Ihnen gewünschte Weise ermöglichen. Autorisierten *Visualizer*-Anwender bietet es die schnelle Aktualisierung von Daten, das Hinzufügen neuer Datensätze und das Ändern von Werten. Diese leistungsfähigen Funktionen und die Benutzerfreundlichkeit verdeutlichen, warum *Visualizer* an erster Stelle unter den OS/2 und DB2 Business Intelligence-Produkten steht. Wahren Sie die Investitionen Ihres Unternehmens in Sie und OS/2 mit *Visualizer* von *ASTRAC*.

Hier einige Highlights von *ASTRAC Visualizer Query* auf einen Blick:

- Jahr 2000 Konformität
- Abfrage- und Suchfunktionen zur Überprüfung Ihrer Daten
- Datenreduzierung in der Datenbank oder lokal in Visualizer
- Fortschrittliche Berichtsfunktionen zum Formatieren und zur Präsentation Ihrer Daten
- Balken-, Linien-, Torten-, Streugrafiken, usw.; 2D oder 3D; X-, Y- und Z-Achsen
- Berechnete Spalten und Datenreduzierung ebenfalls lokal möglich
- Ausgezeichnete Benutzerfreundlichkeit mit umfassender Online-Hilfe und Dokumentation
- Workplace Shell Integration und Drag & Drop-Unterstützung
- Maschinenspezifische DB2-Unterstützung plus AS/400, IXF, DIF, DBF, CSV, usw.
- DB2-Datenbankordner, die Daten mit Zugriffsberechtigungen in Symbolform anzeigen
- Aktualisierung sowie Lesen von Daten

EMPDATA.TAB - Visualizer Table
Table Selected Edit View Help
EmployeeNo 283967
Name Smyth, George
JobCode 415
Division Computing
Sex M F
Department 40
AnnualSalary £17,361.24
Commission £0.00
SalaryType 1
Marital M
BirthDate 13-7-71
Telephone 4004
Education None
Next Previous Add Delete Clear Reset
Subset rows Editing row 4 of the 12 rows chosen out of 200
Subset columns 23 columns chosen out of 23

Abb. 2: Eine Visualizer Tabelle zum Zugriff auf Ihre Daten

- Schnittstelle zum *ASTRAC* Applica - tion System (AS) für VM und OS/390
- Schnittstelle zu Lotus Notes
- Möglichkeit zum Import von IBM QMF-Objekten
- Englische, deutsche und japanische Versionen lieferbar

ASTRAC Visualizer Procedures für OS/2 (VP2v1.3)

Mit *Visualizer Procedures* für OS/2 können Sie die Durchführung Ihrer Abfragen, Berichte, SQL-Anweisungen, usw. automatisieren. Stellen Sie sich folgende Situationen vor:

- Es ist Freitag und Ihr wichtigster Kunde benötigt einen aktuellen Statusbericht.
- Die Rechnungslegung ist abgeschlossen und die Analyse wird sofort fällig.
- Sie sind nicht im Büro, aber Ihr Chef benötigt unbedingt noch heute wichtige Grafiken.
- Sie müssen jeden Tag zur gleichen Zeit einen zusammenfassenden Begriff abgeben.

Salary - Report

Report Selected Edit View Data Help

Date 18-11-98 Page 1

Employee Salary Report

Name	Division	AnnualSalary	RevisedSalary
De Beers, Christopher	Computing	33,305	38,301
Lovell, Jane		18,070	20,780
Rickman, John		37,557	43,191
Smyth, George		17,361	19,965
Loveless, Edith		27,636	31,782
Lister, Andrew		24,259	27,897
Temple, Arthur		21,259	24,447
Jaffe, Kirsty		27,282	31,374
Frost, Kevin		57,398	66,008
Massey, Shirley		18,778	21,595
Butler, Geoff		43,580	50,117
Kilpatrick, Mary		23,385	26,892
	Computing	Average Old	Average New
		29,156	33,529
Bacchus, Judy	Head Office	18,424	21,188
Bailey, Joe		173,612	199,654
Fabrizi, Gina		178,218	204,951
Auerbach, Erica		42,872	49,302
Smith, Fiona		13,110	15,076
Reeves, Karen		110,545	127,127

Red figures indicate revised salary < 22,000

Company Confidential Information

Abb. 3: Visualizer erstellt aus selbst umfangreichen Datensätzen schnell übersichtliche Berichte

In all diesen und vielen anderen Lebenslagen hilft Ihnen *Visualizer Procedures*. Und dies durch einfaches Drag & Drop und ohne Programmiererfahrung.

Eine *Procedure* kann zum Beispiel eine Abfrage für Ihre Firmendatenbank ausführen, die Ergebnisse zur Formatierung in einen Bericht geben, den Bericht an Ihren zentralen Drucker senden, die gleichen Ergebnisse in eine Grafik geben und eine 3D-Grafik erstellen, die Grafik an Ihre Abteilung mailen und schließlich ein Protokoll aktualisieren, das zeigt, das all dies geschehen ist.

Mit dem *Procedure Scheduler* kann eingerichtet werden, daß eine solche Procedure um 6 Uhr morgens an jedem Wochentag ausgeführt wird. Wenn Sie dann am Morgen im Büro eintreffen, wartet der Bericht bereits auf dem Drucker auf Sie, die Grafik ist schon per E-Mail an Ihre Abteilung gesendet worden, und Sie haben nun Zeit, sich um alle anderen wichtigen Aufgaben zu kümmern.

Neben der automatischen Abarbeitung Ihrer *Visualizer*-Objekte können Sie ebenfalls andere Programme mit Hilfe des *Procedure Schedulers* einschließen (.EXE-Dateien, REXX-Skripte, usw.). Daneben können Sie *Procedures* in anderen *Procedures* schachteln, so daß die komplexesten Aufgabenabläufe einfach erstellt, ausgeführt und verstanden werden können.

ASTRAC Visualizer Application Development für OS/2 (VAD2v1.3)

Visualizer Development bietet Ihnen eine für die *Workplace Shell* geeignete Umgebung für das Erstellen maßgeschneiderter Anwendungen - insbesondere mit einer kundenspezifischen Front-End-Anwendung für das Abfragen und Modifizieren von Firmendaten. *Development* besteht aus Desktop-Objekten, die folgendes verwalten:

- Das Aussehen der Benutzeroberfläche

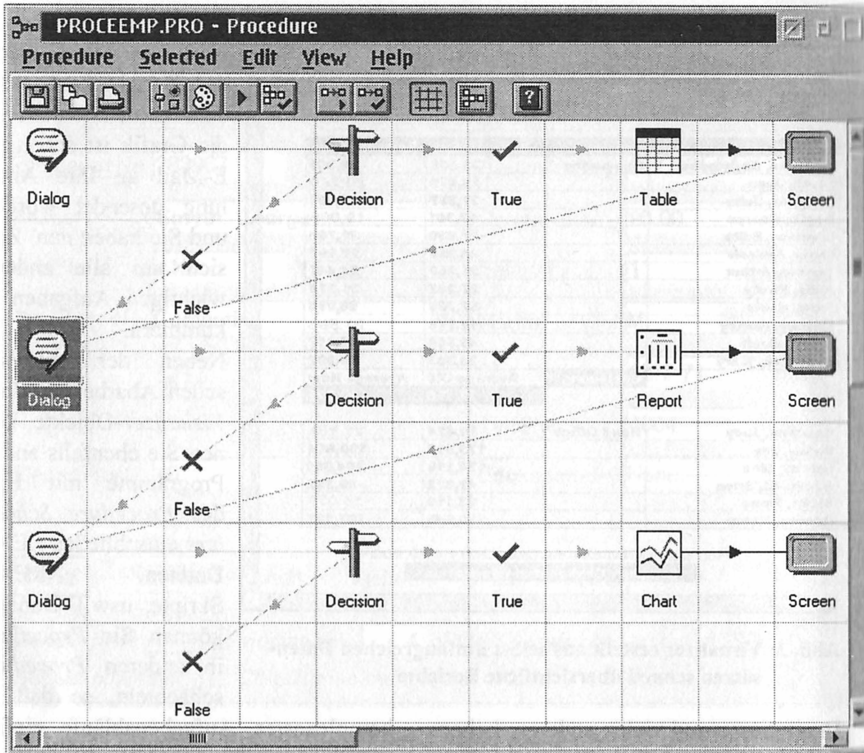


Abb. 4: Visualizer erlaubt das Erstellen von Procedures mit der Maus und damit die Automatisierung selbst komplexer Abläufe ohne Programmiererfahrung.

- Den Inhalt und die Funktionsweise der Aktionsfenstermenüs
- Das Erzeugen und Editieren von Programmcode
- Die Anwendungskomponenten und ihre Kompilierung

Die *Visualizer*-Komponente *Development* verfügt über eine umfassende objektbasierte Programmiersprache (ASL) mit folgenden Eingenschaften:

- Integrierte APIs speziell für *Visualizer Programmodule* wie Bericht und Grafik
- Über 200 Funktionen und 60 Anweisungen
- Fortgeschrittene Debugging- und Tracingfunktionen

- Erstellung und gemeinsame Benutzung Ihrer eigenen Funktionsbibliotheken
- Leistungsfähige Datenanalysefunktionen
- Schnittstellen zu C und OS/2 REXX
- Schnittstelle zum *ASTRAC Application System (AS) für Mainframe*
- Schnittstelle zu Mailsystemen wie *Lotus Notes*
- Multimedia-Objekte zur Integration von Video und Sound
- Hilfe-Manager-Verbindungen zum Erstellen eigener Hilfeoptionen für den Benutzer
- Programmierbare DDE-Verbindungen zu anderen Produkten
- Zugriff auf Programme der DB2-Produktfamilie und nicht-relationale Daten

Anwendungen, die mit *Visualizer Development* erstellt werden, haben vollen Zugriff auf die Leistung und Objekte in der Visualizer Programmfamilie. Dies ermöglicht das schnelle Erstellen maßgeschneiderter Front-Ends unter Verwendung Ihrer Firmendaten, wobei Ihnen *Visualizer*-Programme die harte Arbeit der Analyse und Reduzierung Ihrer Daten abnimmt.

Mit nur wenigen Codezeilen erstellen Sie ein exakt abgestimmtes Front-End mit einem Multimedia-Firmenlog, das Daten aus Ihrer Datenbank extrahiert und dynamisch einen spezifischen Bericht oder eine Grafik erstellt.

ASTRAC Visualizer V5 für OS/2

Neue Funktionen

Das neue Release von ASTRAC Visualizer Version 5 für OS/2 enthält einige der bedeutendsten Erweiterungen in der zehnjährigen Geschichte des Produktes. Die vielfältigen Verbesserungen bieten Lösungen für zentrale Anforderungen, die von *Visualizer*-Anwendern weltweit verlangt wurden und erlauben es ihnen somit, die Vorzüge Ihrer Investition in Datenbank-, Groupware- und Webtechnologien auszuschöpfen. Gleichzeitig begleitet Sie Visualizer sicher in das Jahr 2000 und darüber hinaus. Zu einigen der Erweiterungen in Visualizer V5 gehören:

- Unterstützung der Euro-Währung
- Internet-Ausgabe
- *ASTRAC* AS V5 Unterstützung
- Integration in *Lotus Notes*
- DB2 UDB Ausnutzung
- Erweiterungen bei der
- Anwendungsentwicklung
- Bedienungsfreundlichkeit und Leistung
- Beseitigung und Ausweitung von Grenzen
- Neue Mappingfunktion
- Produktion von Statistiken und Plänen

Euro-Währung

Die Euro-Währung hat Auswirkungen auf Unternehmen, und das nicht nur auf

europäische Häuser. Die Unterstützung für den Euro in *Visualizer V5* wird als eine der besten in der Industrie gerühmt. In *V5* haben Sie simultane Doppelwährungen, mit der eine einfache Migration Ihrer Berichte, usw. möglich ist, und eine zentrale Wechselkurstabelle zur leichten Pflege. Gleichzeitig sind alle leistungsfähigen neuen Funktion in Ihren eigenen *Visualizer*-Anwendungen verfügbar.

Lotus Notes und das Internet

Mit dem Erfolg von *Lotus Notes/Domino* und der verstärkten Integration des Internets in den laufenden Geschäftsbetrieb, können Sie über die neuen Schnittstellen in *Visualizer V5* Ihre wichtigen Berichte und Geschäftsdaten *Notes*- und *Internet*-Benutzern zur Verfügung stellen. Mit *Visualizer V5* können Sie Berichte und Grafiken über Ihre *Lotus Notes* Daten erstellen und sogar Ihre *Visualizer*-Berichte für *Notes*-Benutzer unter Windows NT und anderen Plattformen veröffentlichen. Mit den neuen HTML-, RTF- und GIF-Formaten haben Sie eine Internet-fertige Ausgabe. Das neue Drag & Drop-Retype der *Visualizer*-Objekte ermöglicht eine einfache Verwendung von *Visualizer*-Objekten über E-Mail, usw.

DB2 Universelle Datenbank

Visualizer V5 und die neuesten Releases von DB2 bieten Ihnen nun eine noch bessere Integration zwischen Ihren DB2-Datenbanken und *Visualizer*. *Visualizer V5* hat eine neue Schnittstelle, die UDB API verwendet. Das bietet Unterstützung für DB2 Large Objects für Photos, usw., erhebliche Performanceverbesserungen bei der Arbeit mit großen Tabellen und Unterstützung für 1.000.000.000 Reihen in Ergebnistabellen.

Visualizer Application Development V5

Die vielen Verbesserungen in *Visualizer V5* sind nicht auf *Visualizer Query* begrenzt. Mit *Visualizer Development V5* können Ihre eigenen, kundenspezifischen Anwendungen vollen Nutzen aus diesen Verbesserungen ziehen.

- Verbesserte Dokumentation
- Verbesserte Debugging-Funktion
- Neue Tracing-Funktionen
- Euro EXCHANGE() Funktion
- Unterstützung für neue DB2-Datentypen
- Neuer Programmeditor über EPM
- Neue Schnittstelle zu Lotus Notes
- ...und vieles mehr

Weitere Informationen zum ASTRAC *Visualizer V5* erhalten Sie direkt von ASTRAC:
eMail: sales@astrac.com,
WWW: http://www.astrac.com
Telefon/ Fax: +44-1926-623060/ 623061

Alle Leser der *OS/2 Only!* erhalten übrigens bis zum 31.01.1999 einen Rabatt *von 10 % auf Visualizer und DB2 Lizenzen.* Wenden Sie sich diesbezüglich und für weitere Informationen an die Fa. CONCEPT, dort speziell an Herrn Oliver Raths. Das Firmenprofil der CONCEPT GmbH finden Sie auf der nächsten Seite

CONCEPT Profildaten

Kontakdaten:

Dammtorstr. 31
20354 Hamburg
eMail: vertrieb@CONCEPT-GmbH.de
www: http://www.CONCEPT-GmbH.de
Ansprechpartner: Herr Oliver Raths

Unsere Partnerschaften:

➔ IBM Business Partner

➔ Lotus Business Partner

➔ BusinessObjects Business Partner

➔ ASTRAC Business Partner

Unsere Bereiche:

➔ Consulting

➔ Anwendungsentwicklung

➔ Schulung

➔ Support

➔ Softwareverkauf

Unsere Schwerpunkte:

Anwendungsentwicklung mit:

➔ Lotus Notes / Domino

➔ IBM VisualAge for Java / IBM VisualAge C++

➔ ASTRAC Visualizer

Datwarehousing / business Intelligence mit:

➔ DB2, DB2OLAP Server

➔ IBM VisualWarehouse

➔ BusinessObjects

➔ IBM Datamining

Softwareverkauf:

➔ IBM

➔ Lotus

➔ BusinessObjects und

➔ ASTRAC Softwareprodukte

Aktion:

➔ DB2 Universal Database

➔ ASTRAC Visualizer ❑

In eigener Sache

Gleich zu Beginn des Erscheinens unseres neuen Sammelwerkes gibt es einige Ankündigungen, die wir Ihnen in diesem Abschnitt mitteilen möchten. Wir halten Sie mit diesem Kapitel auch in Zukunft stets auf dem laufenden, wenn wir neue Projekte planen oder sich Änderungen für bestehende ergeben sollten. Auch immer wieder an uns gerichtete Fragen werden wir hier beantworten, da die Antwort für Sie umfassender ausfällt, und wir effizienter mit Ihren Anliegen umgehen können.

Die OS/2 Only! im Abonnement
Viele unserer Leser wollten die *OS/2 Only!* bereits ab dem 1. Exemplar abonnieren. Das ist jedoch nicht möglich, weil unser Magazin erst ab Bd. 2 auch im Abonnement erhältlich ist. Jetzt also steht einem Abo nichts mehr im Wege. Hat Sie das neue Magazin überzeugt, verwenden Sie bitte die beiliegende Bestellkarte, um Ihr Abonnement zu ordern. Bitte haben Sie dafür Verständnis, daß wir Abonnementsbestellungen nur mit dieser Karte bearbeiten können. Sie finden die Karte in jeder Ausgabe der *OS/2 Only!*
Wir bieten Ihnen Jahresabonnements (6 Ausgaben) zu DM 136,92 (Inlandspreis) bzw. DM 147,94 (Auslandspreis). Studenten, Wehrpflichtige, Azubis und Schüler erhalten ermäßigte Abonnements zu 124,99 (Inlandspreis) und 135,07 (Auslandspreis). Zum Vergleich: Eine Einzelausgabe der *OS/2 Only!* kostet DM 24,80 (Inlandspreis) bzw. DM 26,80 (Auslandspreis). Natürlich beliefern wir Sie frei Haus, die oben genannten Preise verstehen sich also inkl. 16 % MwSt., Porto und Verpackung. Einzelheiten zur Abonnementsvereinbarung entnehmen Sie bitte der Bestellkarte. Und falls Sie noch Fragen haben, rufen Sie uns einfach an (die Kontaktmöglichkeiten finden Sie im Impressum am Anfang des Bandes). Übrigens: Ermäßigte Abonnements bekommen auch Vereine, öffentliche Einrichtungen und Benutzerstammtische (ab 5 Mitgliedern).

Die OS/2 Only! CD
Wir wurden schon oft darauf angesprochen, ob es auch CDs zur *OS/2 Only!* geben wird. Ja, CDs wird es geben! Die CDs erscheinen halbjährlich, einmal im Juni zusammen mit der 3. Ausgabe eines Jahrgangs; und im Dezember, gemeinsam mit der Ausgabe 6. Auf den CDs werden enthalten sein: Alle aktuellen Fixpaks für OS/2 und andere OS/2-Produkte von IBM; unsere Treibersammlung, jede Menge Share- und Freeware, Tools zum Programmieren, Quelltexte, Beispielprogramme, Schriftarten, Icons, Bitmaps, Klangdateien, Informationsmaterial zu OS/2 in HTML-Dokumenten und die aktuelle Try & Buy-Version unseres Softwarepaketes *Xelia* zusammen mit kostenlosen Updates für alle *Xelia*-Anwender. Zusätzlich auf der Dezember-CD befindet sich der gesamte Jahrgang der *OS/2 Only!* inkl. Händerlisten, Kontaktadressen, allen Artikeln, einer Datenbank mit Tips & Tricks usw. Auf die CD greifen Sie mit einem speziell für diesem Zweck entwickelten Viewer zu (unter *Xelia* erfolgt der Zugriff über das *Multimedia-CD-ROM-*

OS/2 Only!

Das fortlaufende Sammelwerk nur für OS/2

Abonnement

☐ Hiermit bestelle ich die OS/2 Only! ab der nächsten verfügbaren Ausgabe im Abonnement zum Preis von DM 136,92 (DM 147,94 Auslandspreis) inkl. 16 % MwSt., Porto und Verpackung für 12 Monate (6 Ausgaben). Außerdem erhalte ich als Abonnent der OS/2 Only! auf alle OS/2 Only! CD-ROMs einen Preisnachlaß von 10 %. Das Abonnement ist jederzeit mit einer Kündigungsfrist von 12 Wochen kündbar. Es verlängert sich nach Ablauf eines Jahres, sollte ich nicht kündigen, automatisch um ein weiteres Jahr.

☐ Ich bin Schüler/in, Student/in, Auszubildende/r oder Wehr- bzw. Zivildienstleistender und bestelle das oben beschriebene Abonnement zu einem ermäßigten Preis von DM 124,99 (DM 135,07 Auslandspreis). Diese Karte sende ich daher zusammen mit einem entsprechenden Nachweis ein.

Die Abonnementsgebühren sind gegen Rechnung innerhalb von 14 Tagen nach Erhalt ohne Abzug zahlbar.
Ich kann diese Abonnementsvereinbarung innerhalb von 10 Tagen beim C.-E. Fischer Buchverlag, Stuttgart, schriftlich widerrufen. Zur Wahrung dieser Frist genügt die rechtzeitige Absendung meines Widerrufs (Datum des Poststempels).

☐ Bitte informieren Sie mich außerdem über Ihre Neuerscheinungen aus dem Bereich OS/2 und über Software bzw. Sondererscheinungen zur OS/2 Only!

☐ Ich bin daran interessiert, auch von weiteren Anbietern Informationen -ganz speziell zum Themenbereich OS/2- zu erhalten.

X

Datum, Unterschrift



Xelia für OS/2 ab 2.1

Vormerkung

Xelia, das ist die neue Anwendungs- und Entwicklungsarchitektur nur für OS/2 und beinhaltet über 25 integrierte Anwendungen sowie eine neue graphische Benutzeroberfläche, schön, schlank, schnell und - einfach zu bedienen! Xelia bietet eine konsequent objektorientierte Arbeitsweise, besitzt Multidesktopfähigkeiten und bietet eine ideale Architektur zur Systempflege sowohl von Einzelplatzrechnern als auch von Netzwerken. Das Programmpaket macht dabei intensiven Gebrauch von OS/2-Spezifika. Im Zusammenhang mit Xelia bieten wir ab dem Jahr 2000 auch das XeliaNetwork, einen neuen OS/2-Onlinedienst für alle Xelia-Anwender mit Software, neuen Treibern und Informationen -zum kostenlosen Download-, Serverdiensten und Internetzugang. Vorgemerkte Anwender erhalten regelmäßig Informationen, 15 % Rabatt auf alle Xelia-Produkte und Updates ab dem Zeitpunkt der Vormerkung!

☐ Hiermit lasse ich mich für die nächste verfügbare Version von Xelia vormerken. Als vorgemerkter Kunde erhalte ich ab dem Zeitpunkt meiner Vormerkung die nächste Version und alle folgenden Updates -wie oben angegeben- mit einem Rabatt von 15 % auf alle Preise. Das gilt auch für Produkterweiterungen und Xelia-Anwendungen, neue Treiber sowie sämtliche Online-Dienste, sofern sie im Zusammenhang mit Xelia angeboten werden (XeliaNetwork). Ich bin mit meiner Vormerkung nicht zur Bestellung oder Abnahme verpflichtet.

X

Datum, Unterschrift

Name, Vorname, Titel

Firma, Abteilung

Straße, Hausnummer

Land, PLZ, Ort

Telefon, Telefax

eMail

Bitte mit
DM 1,00
freimachen

C.-E. Fischer Buchverlag
z.Hd. Herrn Ropielewski
Wegländerstr. 24

70563 Stuttgart

Name, Vorname

Firma, Abteilung

Straße, Hausnummer

Land, PLZ, Ort,

Telefon, Telefax

eMail

Bitte mit
DM 1,00
freimachen

C.-E. Fischer Buchverlag
z.Hd. Herrn Ropielewski
Wegländerstr. 24

70563 Stuttgart

Dokument direkt auf dem *Xelia-Schreibtisch*). Dieser Viewer ermöglicht nicht nur das Betrachten der einzelnen Artikel, sondern bietet auch einen komfortablen Zugriff auf die Datenbanken der CD sowie eine einfache Installation bzw. Deinstallation von Programmen, Hilfe zum Aufspielen von Fixpaks und Treibern usw. Alle Texte finden Sie auf der CD übrigens im HTML-Format - versteht sich.
Die CDs gelten als Sondererscheinungen, sind also nicht im Abonnement oder als Beilagen zu den Ausgaben 3 und 6 eines Jahrgangs erhältlich. Bestellkaren liegen der *OS/2 Only!* rechtzeitig bei, und natürlich wird auch eine Bestellung über das Web möglich sein. Sind Sie *OS/2 Only!-Abonnement*, erhalten Sie auf die CDs -wie auf alle anderen Sondererscheinungen zur *OS/2 Only!*- 10 % Rabatt. Der Preis für die CDs steht allerdings noch nicht fest. Wir sind jedoch auf jeden Fall bemüht, ihn nicht über DM 30,00 pro Stück steigen zu lassen.

Offizielles zu Xelia
Uns haben in letzter Zeit immer mehr Anfragen zu *Xelia* erreicht. Auch hier besteht die Befürchtung, das Projekt sei bereits eingestellt, weil es nichts zum Testen gibt. Eingstellt wurde aber auch hier nichts, ganz im Gegenteil.
Wir haben Veränderungen an der Architektur vorgenommen und Anregungen von Interessenten in das Projekt aufgenommen. Die ganze Oberfläche wurde verändert und teilweise auch das Bedienkonzept der neuen Arbeitsoberfläche. Das brauchte und braucht aber noch etwas Zeit. Wir überprüfen ein Programm lieber doppelt und dreifach, ehe Sie es testen können.
Dieses Jahr wird es aber endlich soweit sein. Wir geben Ihnen einen Einblick in eine *Xelia* Version, die speziell zum Testen angefertigt wurde: *Xelia 0.82*. Die Version 0.82 ist eigentlich eine nur für den internen Gebrauch bestimmte Version und beinhaltet viele kosmetische Veränderungen der Oberfläche nicht, genauso wenig wie den REXX-Support zur Makrodefinition und Anwendungsentwicklung, bis auf wenige (aber wichtige!) Ausnahmen keine der *Xelia*-Anwendungen, und auch einige Funktionen des Architekturkernels fehlen, die erst in der Version 0.85 auftauchten. Wenn Ihnen aber die Version 0.82 gefällt, werden Sie sicher auch die Version 1.0 testen wollen, die auf einen Schlag eine Vielzahl neuer Funktionen und Anwendungen bringt. Die Version 0.82 liegt der Juniausgabe der *OS/2 Only!* mit einer ausführlichen Broschüre zur Installation bei und wird auch auf der ersten CD enthalten sein. Außerdem können Sie die offiziellen und kostenlosen Updates von unserer Homepage herunterladen. *Xelia 1.0* wird im Oktober verfügbar sein, und Ende dieses Jahres erhalten Sie das kostenlose Update auf 1.1. Die Version 1.2 folgt im Sommer 2000 und ermöglicht Ihnen den Zugriff auf unseren Informationsdienst *XeliaNetwork*.
Xelia 0.82 dürfte Ihnen Freude bereiten, weil Sie eine Alternative zu *Netscape* bekommen. *Xelia* bietet Ihnen einen integrierten Browser, der mit den aktuellen HTML-Standards umgehen kann (Frames stellen dann keine Hürden mehr dar). Wichtiger ist aber, daß der Browser

gerade einmal 250 KByte groß und damit auch angenehm schnell ist. Dieser Browser wird neben kleineren Anwendungen (Datenbank, HTML-Editor, Verktorgraphikbearbeitung) die erste große *Xelia*-Applikation sein, die in den Schreibtisch integriert sein wird, und demonstriert sehr gut die Leistungsfähigkeit eines Programms, daß auf dem *Xelia*-Konzept basiert. Ausprobieren dürfte sich also lohnen (zumal es ja nichts kostet). Wenn Sie sich für *Xelia* vormerken lassen, schicken wir Ihnen aktuelle Informationen automatisch zu, was auch für kostenlose Updates gilt. Eine Karte zur Vormerkung finden Sie in dieser Ausgabe.

Xelia Ideenwettbewerb

Schreiben Sie uns, was Sie sich mit *Xelia* wünschen, was die neue Arbeitsoberfläche bieten sollte, welche Anwendungen Sie bräuchten und auch, welche Features Sie für unnötig halten. Als Belohnung verlosen wir unter allen Einsendungen 5 Xelia-Lizenzen mit kostenloser Updateberechtigung bis zur Version 2.0 (einschließlich) und 3 Jahresbonnements der *OS/2 Only!* Machen Sie also einfach mit. Ihre Ideen können Sie uns per eMail, Fax oder Post schicken. Einsendeschluß ist der *1. September 1999*. Allein Ihre Einsendung zählt, auch wenn Ihre Ideen bereits von anderen Anwendern bei uns eingegangen sind.

Entwickeln mit Xelia

Wenn Sie Programmierer sind, machen Sie Ihre Anwendungen Xelia-tauglich. Damit sparen Sie sich eine Menge Routinearbeit. Ab Herbst erhalten Sie für *Pascal* und *C* kostenlose Toolkits zur Xelia-Anwendungsentwicklung. Die Oberfläche Ihres Programms entwickeln Sie dabei visuell mit der Maus über eine IDE, die mit der Version 1.0 ausgeliefert wird. Mit *Xelia* 1.1 können Sie auch TCP/IP-Anwendungen entwerfen und Ihre Programme damit auf einfachem Wege netzwerktauglich machen.

Wenn Sie nur mit REXX entwickeln, benötigen Sie kein spezielles Toolkit. *Xelia* eröffnet für alle eine einfache Möglichkeit der Anwendungsentwicklung. Die in *Xelia* integrierte Entwicklungsumgebung benutzt REXX zur Applikationserstellung. Wie das funktioniert, können Sie dem Handbuch zu *Xelia* und der Onlinedokumentaion entnehmen.

XeliaNetwork

Xelia ist mehr als nur ein Softwarepaket. Die neue Oberfläche eröffnet Ihnen auch einen neuen Zugang zu Online-Ressourcen. Für das Jahr 2000 planen wir das *XeliaNetwork*, ein Netz, auf das Sie als *Xelia*-Anwender mit einem speziellen Programm, das auf einem architekturinternen Protokoll basiert, auf unsere Server zugreifen können. Das *XeliaNetwork* bietet Ihnen wichtige Dateien und Informationen zu OS/2, Software, Fixpaks und vieles mehr zum kostenlosen Download. Daneben erhalten Sie auch eine eMail-Adresse, Zugang zum Internet und Platz für Ihre Homepage. Auf Internetressourcen greifen Sie ebenfalls über das *XeliaNetwork* zu.

Das Netzwerk wird bundesweit zum Ortstarif sowohl über Modem als auch ISDN und über das Internet erreichbar sein. Sie profitieren übrigens auch mit einer

Xelia-Vormerkung von Rabatten zum *XeliaNetwork.*

Die jeweils aktuellen Informationen zu unserem Onlinedienst erhalten Sie auf unserer Homepage. Daneben informieren wir Sie als Leser der *OS/2 Only!* regelmäßig über den Planungsstatus des Netzes.

Vorschau auf die nächste Ausgabe

Die Themen der nächsten Ausgabe stehen bereits fest. Änderungen sind aus aktuellen Anlässen möglich. Natürlich nehmen wir Ihre Artikel trotz bestehender Themen in eine Ausgabe auf oder nehmen am Aufbau einer Ausgabe entsprechende Änderungen vor, wenn uns Ihr Beitrag besonders wichtig erscheint. Die Themen der *OS/2 Only! Bd.2* sind:

Neuigkeiten

Enthalten sein werden Kurzmeldungen und unsere erste Treiber- und Programmübersicht.

Hardware

Praxis: CD-ROM-Recorder

Wir testen verschiedene CD-ROM-Recorder, die sich für den Einsatz unter OS/2 eignen.

Software

Praxis: RSJ CD-Writer

In Zusammenhang mit dem Hardwarekapitel stellen wir Ihnen den RSJ-CD-Writer vor.

Konfigurieren und Anwenden

Praxis: Wartungspartitionen einrichten

In diesem Aritkel zeigen wir Ihnen, wie man Wartungspartitionen einrichtet und für den Notfall entsprechend konfiguriert.

Theorie: Die OS/2 DOS-Unterstützung

Ein umfassender Artikel, der zeigt, wie OS/2 die Unterstützung für DOS- und Windows-Programme zur Verfügung stellt. Mit Tips zur Konfigurationen für DOS- und Windows-Anwendungen.

Programmieren

Praxis: Workshop DLLs, Teil 2

Entwickeln von Bibliotheken für REXX und eines REXX-Makrosupports für das Beispielprogramm.

Netzwerk

Praxis: HTML Workshop, Teil 2

Der zweite Teil unseres HTML-Workshops mit Hilfsmitteln zum Webdesign und einer ausführlichen Sprachreferenz.

Theorie: Grundlagen zu TCP/IP

Zur Vorbereitung unseres Workshops *Netzwerke mit OS/2* geben wir Ihnen das nötige Basiswissen zu TCP/IP in die Hand.

Außerdem in den nächsten Ausgaben: ***Hardware:*** *SCSI-Workshop* für den leichten Umstieg von IDE auf SCSI; *OS/2-Rechner selbst zusammenstellen* für den Aufbau OS/2-tauglicher Computersysteme. ***Software:*** *Netscape Communicator* und *Astronomiesoftware für OS/2*; ***Netzwerk:*** *Netzwerke mit OS/2* zum Aufbau kleiner Peer- und TCP/IP-Netzwerke in unterschiedlicher Ausführung; *Mailboxen für OS/2* zur Konfiguration eigener Mailboxsysteme und natürlich vieles andere mehr. ❑

Stichwortverzeichnis

A

Adreßbus: Das Adressieren des Speichers wird über Adreßleitungen realisiert, Leiterbahnen, die vom Prozessor zum Systemspeicher reichen. Die Gesamtanzahl dieser Adreßleitungen bezeichnet man als den *Adreßbus*. Am Prozessor selbst ist der Adreßbus definiert durch die Anzahl der für die Adressierung vorgesehenen Pins am CPU-Gehäuse.

Adressieren, Adreßräume: Wenn wir von Adressierung sprechen, so meinen wir das Ansprechen einer Stelle im Arbeitsspeicher des Systems. Durch den binären Charakter eines Prozessors kann dieser mit 1 Adreßleitung 2^1 Speicherstellen ansprechen oder -allgemeiner ausgedrückt- mit n Adreßleitungen vermag der Prozessor auf 2^n Speicherstellen zu adressieren. Dabei verhält sich die Hardware linear, d.h. zu allen 2^n Kombinationen gibt es genau ein Speicherelement, jeweils ein Bit. So spricht der i8086 den Speicher mit 16 Bit (16 Adreßleitungen), der i80386 mit 32 Bit (32 Adreßleitungen) an. Damit kann der i8086 2^{16} Bit, also 64 KByte, der i80386 2^{32} Bit, also 4 GByte Speicher adressieren. Die Grundeinheit der Adressierung ist dabei aber das Byte, d.h. die Adresse, die z.B. 5 heißt, weist auf den Anfang des fünften Bytes im Speicher und nicht auf das fünfte Bit. Den Speicher, den ein Prozessor mit seinen Adreßleitungen verwaltet, bezeichnet man als *physikalischen Adreßraum*, Eben weil zu jeder Adresse (oder 2^n-Kombination) genau ein Speicherelement existiert. Auf diesen Adreßraum kann der Prozessor physikalisch, also tatsächlich zugreifen. Installiert man in einem 80386 System 4 GByte RAM, so könnte der Prozessor jedes Bit dieses Speichers addressieren, jedoch kein einziges Bit mehr. Damit sind 80386-kompatible CPUs auf einen maximalen Hauptspeicherausbau von 4 Gbyte beschränkt.
Daneben gibt es noch den *logischen Adreßraum*. Beim logischen Adreßraum greift ein Programm auf einen sehr viel größeren Speicherbereich zu, der durch den Prozessor auf den physikalischen Adreßraum abgebildet wird. Die logischen Adressen bezeichnen also keine tatsächlichen (physikalischen) Speicherplätze. Um auf ein Speicherelement real zuzugreifen, wird die logische Adresse von der CPU in eine physikalische übersetzt. Im ersten Fall spricht man auch von einem →*linearen*, im zweiten Fall von einem →*segmentierten Speichermodell*. Dies eröffnet den interessanten Aspekt der virtuellen Adressierung, mit welcher der 80386 (im Protected Mode) arbeitet.

B

Bit, Byte: Das Bit ist die kleinste Informationseinheit in einem Rechnersystem. Bit ist ein Kunstwort, zusammengesetzt aus »Binary Digit«, also »Binärziffer«. Gemäß des Dualsystems kann sich ein Bit also als 0 oder 1 präsentieren und gibt damit den Zustand eines elektronischen Schaltvorganges wieder.
Da man mit mehreren Bits arbeitet, werden Bits zu Bytes zusammengefaßt. Ein Byte besteht aus 8 einzelnen Bits. Weitere Informationseinheiten, die man aus Bits zusammensetzt, sind:

1 Wort (Word) = 2 Byte = 16 Bit,
1 Doppelwort (Double-Word)= 4 Byte = 32 Bit,
1 Vierfachwort (Quad-Word) = 8 Byte = 64 Bit.

Wieviele Zustände (zum Beispiel Zahlen) durch ein Byte ausgedrückt werden können errechnet sich einfach, indem man die Basis des zugrundeliegenden Systems -des Dualsystems- mit der Anzahl an Bits potenziert. Mit einem Byte ergeben sich damit 2^8, also 256 Kombinationsmöglichkeiten. Mit jeder Kombination könnten 256 verschiedene Bits im Speicher angesprochen (adressiert) werden.

Immer wenn man eine Zehnerpotenz erreicht, benutzt man die aus dem metrischen Maßsystem bekannten Einheiten, also ^{10} = Kilo, ^{20} = Mega, ^{30} = Giga, ^{40} = Tera usw. 2^{10} Bit sind 1024 Byte, also 1 KByte. Bei der Umrechnung großer Bytewerte ist also nicht 1000 als Quotient zu wählen, wie im metrischen Maßsystem, sondern die durch das Binärsystem bedingten 1024.

C

Coprozessor: Auch →*FPU*. Ein mit der CPU zusammenarbeitender Prozessor, der Gleitkommaoperationen ausführt. Nötig, da die CPU nur Ganzzahlen verarbeiten kann. Coprozessoren arbeiten parallel zur CPU. Ist in einem System keine FPU vorhanden, muß die CPU entsprechende Algorithmen zur Emulation eines Coprozessors ausführen. Unter OS/2 stellt die DLL *NPXEMLTR.DLL* diese Algorithmen zur Verfügung.

D

Deskriptor: Eine vom System benutzte Struktur zur Beschreibung von Segmenten. Auf Deskriptoren hat nur das Betriebssystem Zugriff. In ihnen wir die Basisadresse eines Segmentes un die Zugriffsrechte auf das Segment fest gehalten. Damit sind Deskriptoren El mente der virtuellen Adressierung.

DLL: *Dynamic Link Library*. Bibliothek die Funktionen und Ressourcen enthäl welche zur Laufzeit mit einem Pro gramm verbunden werden.

Datenbus: Die Gesamtheit aller Pins an Gehäuse der CPU, die dazu dienen Daten zwischen dem Prozessor und de Peripherie (Speicher, Erweiterungska ten etc.) auszutauschen. Der Datenbu stimmt meistens mit der internen Vera beitungsbreite überein, da in diesem Fa für jedes im Prozessor verarbeitete B ein Pin zur Verfügung steht, das mit de anderen Systemkomponenten ausge tauscht werden kann.

Dualsystem: Um Informationen zu vera beiten, benötigt man ein Trägersysten (etwa eine Welle zur Schallausbreitung das Tägersystem beim Sprechen). I einem elektronischen System besteht ei solches Trägersystem in Strömen. A technisch einfachsten zu realisieren sin Unterscheidungen zwischen verschied nen Zuständen wie z.B. »Strom fließt Strom fließt nicht«. Zur Datenverarbe tung zweier solcher Zustände bietet sic demnach das Dualsystem an, da es fü die physikalische Realität ein optimal Modell darstellt.

Die Basis des Dualsystems ist die 2, un es existieren 2 verschiedene Ziffern, d 0 und die 1; die ersten zwei Zahlen en sprechen diesen beiden Ziffern. Wi die 1 überschritten, wird eine neu Stelle hinzugefügt, der 1 folgt die 1

wird die 11 überschritten, wird wieder eine Stelle hinzugenommen, der 11 folgt die 100; wird die 111 überschritten, wird erneut eine Stelle hinzugenommen, der 1111 folgt die 10000. Zwischen diesen Zahlen zählt man weiter. Die Reihe der ersten siebzehn Dualzahlen sieht dann wie folgt aus:

0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000 ...

Da wir im Dezimalsystem rechnen, ist es notwenig, Dualzahlen in das Dezimalsystem umzurechnen; für den Computer ist notwendig, Dezimalzahlen, die man eingibt, in das Dualsystem umzurechnen. Für die erste Umrechnung verwendet man das Multiplikationsverfahren: Man rechnet die Dualzahl von rechts in das Dezimalsystem um. Die Stellenwerte der einzelnen Ziffern entsprechen im Dualsystem dabei den Potenzen der Basis 2. Man schreibt die einzelnen Stellen der Dualzahl von rechts beginnend untereinander, ihre Stellenwerte daneben, errechnet das Produkt und addiert die Proukte. Das Ergebnis ist die der Dualzahl entsprechende Zahl im Dezimalsystem. Die Dualzahl 1101 wird damit wie folgt umgerechnet:

$1 \times 2^0 = 1$
$0 \times 2^1 = 0$
$1 \times 2^2 = 4$
$1 \times 2^3 = 8$

13D

Die zweite Umrechnung erfolgt mit dem Divisionsverfahren: Man teilt die Dezimalzahl durch die Basis des neuen Systems bis man das Ergebnis 0 erhält. Man schreibt jeden Divisionsvorgang mit den entstehenden Resten untereinander. Die Reste der einzelnen Divisionen von unten nach oben gelesen ergeben die gesuchte Zahl im Dualsystem. Die Dualzahl 32 wird z.B. so umgerechnet:

32 : 2 = 16	Rest 0	
16 : 2 = 8	Rest 0	
8 : 2 = 2	Rest 0	
2 : 2 = 1	Rest 0	
1 : 2 = 0	Rest 1	10000B = 32D

Um die Zahlen eindeutig voneinander unterscheiden zu können, versieht man Dualzahlen wie oben angegeben mit einem B, Dezimalzahlen mit einem D.

Dynamic Linking: Vorgang, bei dem Programmcode oder Ressourcen während der Laufzeit eines Programms und nicht zum Zeitpunkt des Compilierens mit diesem Programm verbunden werden.

E

Exception: Ausnahmebedingung. Exceptions werden durch Verletzungen der Schutzmechanismen des Prozessors (z.B. Zugriffe auf Daten höherer →*Privilegebenen* von niederen Privilegebenen aus; Zugriffe auf nicht existierende Speicherbereiche) oder Rechenfehler (z.B. Division durch 0) ausgelöst. Zur Behandlung dieser Ausnahmebedingung dient ein sog. *Exception Handler*.

F

FAT: s. Bd. 2.

Flat Memory Model: Auch *flaches* oder *lineare Speichermodell* Speichermodell,

das zur Adressierung des Speichers keine Segmente mehr verwendet, sondern den Adreßraum als ein zusammenhängendes Kontinuum darstellt, dessen Größe von der Adreßbusbreite des Prozessors abhängig ist. OS/2 stellt seit der Version 2.0 ein Flat Memory Model zur Verfügung.

FPU: Abkürzung für *Floating Point Unit*. Siehe: →*Coprozessor*.

G

GDTR: *Global Deskriptor Table Register*. Register des 80386, das den Deskriptor der *Global Deskriptor Table* aufnimmt. Da die GDT in einem Multitaskingsystem nur einmal vorhanden ist, wird der Deskriptor in einem Resgister des Prozessors, nicht in einer Tabelle gespeichert.

H

HPFS: s. Bd. 2.

Hyperlink: Eine in einem Online-Dokument farblich hervorgehobene Textstelle oder eine Graphik, die auf ein anderes Dokument, eine Datei oder (bei HTML) auf einen anderen Server verweist. Der Benutzer gelangt durch einen Mausklick auf den Link auf die durch diesen Link referenzierten Daten.

Hyper Text Markup Language: Abkürzung HTML. Standardisierte Tag-Sprache für Webdokumente. HTML-Dateien sind normale Textdokumente. Die zur Formatierung des Dokumentes festgelegten Tags mit ihren Argumenten verwendet ein sog. Webbrowser, um die Dokumente in einer visuell ansprechenden und übersichtlichen Form anzuzeigen. Es existieren mehrere Sprachstandards von HTML.

I

ISA-Bus: s. Bd.2.

IDTR: *Interrupt Deskriptor Table Register*. Register des 80386, das den Deskriptor der *Interrupt Deskriptor Table* aufnimmt. Da die IDT in einem Multitaskingsystem nur einmal vorhanden ist, wird der Deskriptor in einem Resgister des Prozessors, nicht in einer Tabelle gespeichert.

L

LDTR: *Local Deskriptor Table Register*. Register des 80386, das den Deskriptor der gerade aktuellen *Local Deskriptor Table* aufnimmt. Das LDTR wird bei jedem Taskwechsel neu geladen, da zu deisem Zeitpunkt

Lineares Speichermodell: Siehe →*Flat Memory Model*.

M

Micro Channel Architecture: s. Bd. 2

N

NLS: s. Bd. 2

P

Page: Auch Speicherseite. Ein 4 KByte großer Speicherbereich. Ist das →*Paging* des Prozessors aktiv, wird der

gesamte Speicher in Pages unterteilt. Unter OS/2 ist eine Page die kleinste allokierbare Speichereinheit.

Page-Directory: s. →*Page-Directory-Table.*

Page-Directory-Table: Eine speicherresidente Tabelle, deren Einträge (*Page Directory Entries PDEs*) die Basisadressen der im System vorhandenen Page-Tables beinhalten. Die PDEs haben den gleichen Aufbau wie die Page-Table-Entries.

Page-Table: In Anlehnung an die UNIX-Konvention können sowohl die *Page-Tables* als auch die *Page-Directory-Tabelle* als Verzeichnisse aufgefaßt werden. Das heißt, daß in den Page-Tables die Namen der Pages enthalten sind und die Page-Directory-Tabelle die Namen aller Page-Tables beinhaltet. Man nennt die Page-Directory-Table auch Wurzel-Verzeichnis.

Page-Table-Deskriptor: Ein in der Page-Directory-Tabelle eines Systems eingetragener Deskriptor, der sich aus zwei Feldern zusammensetzt: Das erste Feld (Bit 0-11) enthält statistische Informationen und die Zugriffsrechte einer individuellen Page-Table. Das zweite Feld (Bis 12-31) enthält die obersten 20 Bits der 32-Bit physikalischen Basis-Adresse einer individuellen Page-Table.

Page-Deskritor: Deskriptoren, die als Einträge in einer Page-Table dienen. Die in einer Page-Table enthaltenen Page-Deskriptoren haben das gleiche Format wie die Page-Table-Deskriptoren: Das erste Feld (Bit 0-11) enthält statistische Informationen und die Zugriffsrechte einer individuellen Page, das zweite Feld (Bis 12-31) enthält die obersten 20 Bits der 32-Bit physikalischen Basis-Adresse einer individuellen Page.

Paging: Ein neben dem Segment-Übersetzungsmechnismus des 80386 optional aktivierbarer Adreßübersetzungsmechanismus. Die von der Segmentübersetzung gelieferten linearen Adressen werden dann dem nachgeschalteten Paging-Mechanismus übergeben, welcher sie in physikalische Adressen übersetzt. Beim Paging wird der Systemspeicher in 4 KByte große Abschnitte, den Pages unterteilt.

Pipelining: Ein Rechnersystem arbeitet prinzipiell mit zwei Datentypen: Befehle und Operanden. Befehle sind Instruktionen, die angeben, welche Operationen mit den Operanden durchgeführt werden sollen. Ein Prozessor wie der 80386 kann geichzeitig mehrere Befehle abarbeiten, genauer: Innerhalb einer Ausführungszeit, z.B. zwei Takten, kann der Prozessor Operationen, die zur Ausführung notwendig sind, schon beim vorangegangenen Befehl bearbeiten. Damit führt eine CPU, der über die Fähigkeit des Pipelinings verfügt, einen Befehl etwa zweimal so schnell aus wie ein Prozessor, welcher seine Instruktionen sequentiell abarbeitet.

PM: Abkürzung für *Presentation Manager.* Der PM wurde mit OS/2 1.1 eingeführt und bietet eine vielseitige, nachrichtenbasierende Architektur, die eine Ansammlung mehrer DLLs darstellt, welche den OS/2 Kernel zu einem einfach zu bedienenden GUI-System erweitern. Der Presentation Manager ist die primäre OS/2-Anwendungsumgebung

Privilegebenen: Eine innerhalb der 80386-Schutzmechanismen definierte Abstufung von Zugriffsrechten für Segmente. Der Prozesor bietet vier Privilegebenen, abgestuft von 0 bis 3, wobei 0 die am meisten privilegierte und 3 die am nied-

rigsten privilegierte Ebene ist. Durch die Privilegebenen erhalten Programme verschieden abgestufte Zugriffsrechte auf Systemressourcen.

Protected Mode: Betriebsart des 80386, in der Programme nur mit virtuellen Adressen arbeiten, welche vom Prozessor in physikalische Adressen übersetzt werden. Dadurch sind die Adreßräume der einzelnen Tasks voneinander geschützt, wodurch dieser Betriebsmodus seinen Namen erhält. Im Protected Mode sind ferner alle Schutzprinzipien des Prozessors aktiv.

R

Real Mode: Betriebsart des 80386. Der Prozessor läuft nach dem Einschalten oder einem Systemreset in diesem Modus und verhält sich wie ein 8086. Dabei entsprechen die in den Segmentregistern geladenen Segmentadressen tatsächlichen physikalischen Adressen. Programme benutzen damit ebenfalls physikalische Adressen, woher dieser Betriebsmodus seinen Namen erhält. Im Real Mode sind die Schutzmechanismen und die virtuelle Speicherverwaltung des Prozessors nicht verfügbar.

Register: Ein Register ist ein im Prozessor eingerichteter, also interner Speicherplatz, der je nach CPU-Modell unterschiedlich groß ist. Computersysteme arbeiten mit zwei Datentypen: Befehlen und Operanden. Ein Befehl ist eine im Prozessor festgelegte Definition, die angibt, welche Operationen auf den Operanden ausgeführt werden. Operanden sind nun nichts anderes als Zahlenwerte im Binärformat. Um sie verarbeiten zu können, müssen sie in der Recheneinheit des Prozessors abgelegt werden - eben den Registern. Register sind demna Platzhalter.

S

Segment: Die kleinsten zusammenhänge den Regionen des Speicehrs mit eigen Schutz-Attributen. Segmente geben d funktionellen Aufbau eines Program wieder. So existieren für ein Modul Code-Segment, ein Daten-Segment u für die Task ein Stack-Segment. Jed Segment wird durch einen Deskrip angesprochen. Zugriffe auf Segmen die nicht der Definition des Deskript entsprechen, können daher von Schutzhardware verhindert werden.

Segmentiertes Speichermodell: Speich modell, bei dem der Systemspeicher →*Segmente* unterteilt wird, die e variable Länge von 1 Byte bis 4 GB haben können. Bei diesem Speicherm dell besteht ein Programm damit verschiedenen Segmenten. D erschwert die Anwendungsentwicklu und den erhöht den Verwaltungsa wand für das Betriebssystem.

Selektor (Segment-Selektor): Ein 16-B Zeiger, der als Index für den Zugriff Deskriportabellen dient und im segm tierten Speichermodell für jede Sp cherreferenz benötigt wird. Der Sel tor ist in 3 Felder unterteilt:

- ❑Index-Feld,
- ❑Table-Indikator-Feld (TI),
- ❑RPL-Feld (Requestor's Privilege Level)

Zur Auswahl eines Segmentes wird das Index-Feld und der Table-Indika

benutzt, womit 14 Bits (Bit 2 bis 15) zur Festlegung des virtuellen Adreßraumes zur Verfügung stehen; ein Programm kann damit aus 2^{14} = 16384 unterschiedlichen Segmenten bestehen, wobei ein Segment eine varibale Länge bis zu 4 GByte besitzt. So entsteht ein virtueller Adreßraum von bis zu 64 Terabyte. Durch den Table-Indikator wird der Adreßraum eines Programms in zwei Bereiche unterteilt:

❑globaler Adreßraum (wenn TI = 0), der von Systemdaten- und Prozeduren benutzt wird, und für alle Tasks ver fügbar sein kann.

❑lokaler Adreßraum (wenn TI = 1), der für jede Task definiert ist, womit Code und Daten privat und daher vor dem Zugriff anderer Tasks geschützt sind.

Mit diesem Selektor wird nun ein Deskriptor aus einer Deskriptortabelle herausgesucht. Ein Segment-Desktriptor dient zur Definition eines Segmentes. Dieser Deskriptor besteht aus 8 Bytes, in denen folgene Informationen enthalten sind:

❑BASE, ein 32-Bit-Feld in den Bytes 2, 3, 4 und 7 des Deskriptors, das die physikalische Basisadresse des Segmentes angibt,

❑ACCESS RIGHTS, ein 8-Bit-Feld in Byte 5 des Deskriptors, das Informationen für die Schutzprinzi pien im Protected Mode speichert,

❑LIMIT, 20-Bit-Feld in den Bytes 0, 1 und 6 des Deskriptors, welches die Obergrenze eines Segmentes im Bereich von 2^{20} Bytes = 1 MB festlegt.

❑FLAGS, ein 4-Bit-Feld mit folgen den Elementen:

o zwei 0-Bits für die Kompatibilität mit zukünftigen Prozessoren,

o ein Granularity-Bit G, das dazu dient, die physikalische Adressierbarkeit eines Segmentes auf 4GByte zu erweitern. Dieses Bit bestimmt die Maßeinheit, die der Festlegung der Größe eines Seg mentes zugrunde gelegt wird, womit direkt die Interpretaton des LIMIT-Feldes beeinflußt wird. Ist G=0, ist das Byte die Maßeinheit, um die Größe des Segmentes zu bestimmen.

o ein Default Operation Size-Bit D, das die Länge eines Segmetes bestimmt und daher mit den ASM386 USE-Attributen korrespondiert: Ist D=1, wird ein USE32-Seg ment spezifiziert, ist D=0 ein USE16-Segment.

Da das LIMIT-Feld die Obergrenze eines Segmentes festlegt, bestimmt der Wert in diesem Feld, wieviele Bytes das Segment enthält, womit maximal 2^{20} Bytes · 1 Byte = 2^{20} Bytes = 1MByte pro Segment adressiert werden können. Ist G=1, wird das Segment in Pages unterteilt. Eine Page ist 4 KByte (2^{12} Bytes) groß und wird als Maßeinheit zur Größenbestimmung des Segmentes herangezogen.
Da das LIMIT-Feld die Obergrenze eines Segmentes festlegt, bestimmt der Wert in diesem Feld, wieviele Pages das Segment enthält, womit maximal 2^{20} Pages · 2^{12} Bytes = 2^{32} Bytes = 4 GByte pro Segment adressiert werden können.

SOM: s. Bd. 2.

Stack: Ein Stack ist eine Speicherregion, die zum Sichern von Registerinhalten benutzt wird. Das ist notwendig, wenn ein Programm ein Unterprogramm aufruft. In einem solchen Fall wird der Ablauf des Hauptprogramms unterbrochen, um das Unterprogramm (Prozedur) auszuführen. Nachdem die Prozedur beendet ist, wird mit dem Ablauf des Hauptprogramms fortgefahren. Um eine Prozedur ausführen zu können, muß dieses Programm natürlich die Register des Prozessors belegen, d.h. alle Registerinhalte, welche die Operanden des Hauptprogramms vor dem Aufruf des Unterprogramms beinhalteten, müssen für die Dauer der Ausführung der Prozedur gesichert werden, ansonsten gingen sie verloren, und das Hauptprogramm könnte später nicht fortgesetzt werden.

Daher werden die Registerinhalte auf den Stack geladen. Wird eine weitere Prozedur innerhalb des zuvor aufgerufenen Unterprogramms aufgerufen, werden die Registerinhalte der ersten Prozedur ebenfalls auf den Stack gelegt. Die einzelnen Daten werden also übereinandergestapelt, weshalb man einen Stack auch Stapelspeicher nennt.

Ist eine Prozedur beendet, wird stets der letzte Eintrag im Stack (TOS) wieder in die Register geladen, womit das vor der Prozedur aktive Programm (ein anderes Unterprogramm oder das Hauptprogramm) fortgesetzt werden kann. Ein Stack ist also ein First-in-Last-Out-Speicher.

Für OS/2-Programmierer ist es interessant zu wissen, wie der Stack durch den 80386, auf dessen Architektur OS/2 ab 2.0 basiert, implementiert wird.

Im allgemeinen legt hierbei das Stack-Segment-Register SS die Basisadresse des Stacks im Speicher fest. 80386-Systeme können eine Vielzahl von Stacks verwalten; welcher aber benutzt wi hängt von dem Wert ab, der im Register SS st womit zu einer bestimmten Zeit immer nur Stack addressierbar ist. Unter OS/2 ist das Sta segment allerdings auf den gleichen virtuel linearen Adreßraum abgebildet, da die Selekt komponente des Segment-Selektors (wie allen anderen Selektoren) 0 ist.

Das Stack-Pointer-Register ESP enthält ei Pointer, der immer auf den letzten Eintrag Stack zeigt, auf den sogenannten Top Of St (TOS). Der Stack wächst in Richtung niederw tiger Adresse, also in Richtung der Basisadres weil der 80386 bei Einträgen in den St (PUSH-Operation) den Wert im ESP mit dekrementiert und bei POP-Operationen ESP 4 inkrementiert.

Das Stack-Frame-Pointer-Register wird schli lich benutzt, um Datenstrukturen, Arbeitsbe che und Variablen innerhalb der zur augenbli lichen Prozedur gehörenden Stackbelegu (Stack-Frame) zu adressieren. Damit werden Stack-Frames der verschiedenen Prozedu gegeneinander abgegrenzt.

Wie groß der Stack beim Start eines P gramms ist, kann vom Programmierer *.DEF (Module Definition File) mit Anweisung STACKSIZE festgelegt w den. Im Normalfall beträgt die Grö eines Stacks 16 Kbyte.

Static Linking: Verfahren, bei dem Funk onscode und Ressourcen zur Zeit Compilierens eines Programms mit d sem Programm verbunden werden. so gelinkte Code bzw. die Ressourc sind dann im EXE-Modul enthalten.

T

Takt: Ein elektronisches Signal, daß eine synchrone Steuerung aller Einhei eines Systems sorgt, man kann ihn a als elektronischen Herzschlag des Re ners bezeichnen. Das Taktsignal w von einem Quarzoszillator als Rechtec

pannung mit einer vorgegebenen Frequenz erzeugt (gemessen in MHz). Befehle arbeitet der Prozessor innerhalb von Takten ab. Ausführungszeiten bei Berechnungen werden demnach in Takten gemessen.

Task: Eine zur Ausführung kommende Befehlssequenz. Im Gegensatz zu einem Programm, das statisch ist und aus einem festen Satz an Code- und Datensegmenten besteht, ist eine Task dynamisch und führt ein oder mehrere Programme aus, wenn sich diese im Adreßraum der Task aufhalten. Ein Programm wird zur Task, sobald das Betriebssystem eine Task für dessen Code bereitstellt.

Task Status Segment: Abkürzung *TSS* . Eine vom Betriebssystem verwaltete Struktur, welche eine Task im Detail beschreibt. Normalerweise existiert für jede Task im System ein TSS. Mit den in dieser Struktur gespeicherten Informationen wird das Wechseln zwischen verschiedenen Tasks ermöglich. OS/2 macht vom TSS-Mechanismus nur geringen Gebrauch und implementiert für ein leistungsfähigeres Multitaskingsystem einen eigenen Task-Scheduler.

TLB: Abkürzung für Translation Lookaside Buffer. Ein in der Paging Unit des Prozessors implementiertes Cache-System zur Erhöhung der Leistung beim Paging. Der TLB besteht aus vier Speicherblöcken mit jeweils 8 Eintragungen. Jede dieser Eintragungen besteht aus:

- einer 24 Bit-TAG-Komponente, welche die oberen 20 Bits einer linearen Operandenadresse mit dem DIR- und TABLE-Feld sowie 4 Attribut-Bits enthält, und
- einer 20 Bit-DATEN-Komponente, welche die oberen 20 Bits zur lineraren Operandenadresse gehörenden Page-Frame-Adresse enthält.

Da der TLB insgesamt 32 verschiedene Page-Frame-Adressen enthält, die jeweils mit einer 4 KByte-Page-Größe gekoppelt sind, liegt der Geltungsbereich des TLB bei 128 KB.

TOS: Top Of Stack. Bezeichnung für den letzten Eintrag eines Stacks, also des Eintrags, der zuletzt auf den Stapelspeicher gelegt wurde.

U

URL: Abkürzung für *Uniform Resource Locator*. Hinter einer URL verbirgt sich die Bezeichnung eines im Internet verfügbaren Dokumentes inkl. der vollständigen Angabe der Internetadresse des Servers, auf dem das Dokument zu finden ist. Eine gültige URL wäre z.B. http://www.server.de/index.html.

V

Verarbeitungsbreite, interne: Die interne Verarbeitungsbreite drückt aus, wieviel Bits ein Prozessor gleichzeitig verarbeiten kann, etwa für Additionen, Subtraktionen, Multiplikationen usw. Beim 80386 beträgt die interne Verarbeitungsbreite 32 Bit. Die interne Verarbeitungsbreite gibt neben der Größe des Adreßbusses ebenfalls an, wieviel Speicher der Prozessor adressieren kann. Wird der Speicher mit 32 Bit angesprochen, muß die CPU auch in der Lage sein, 32-Bit breite Adressen zu verarbeiten. (Grundsätzlich kann man auch 32 Bit breite Adressen mit einem Prozessor generie-

ren, der nur eine interne Verarbeitungsbreite von 16 Bit hat. Weil es aber kein 32-Bit-Register zur Speicherung einer solchen Adresse gibt, muß die Adresse auf zwei 16-Bit-Register verteilt werden, was zu mehreren Lade- und Rechenvorgängen bei der Adressgenerierung und damit zu erheblichen Leistungseinschränkungen führt.)

W

Wort: s. →*Bit, Byte*.

WPS: Abkürzung für *Work Place Shell*. Die WPS ist ein objektorientiertes Benutzer-Interface, das auf dem PM und einem ebenfalls obejktorientierten Anwendungskonzept, dem →*SOM* basiert. Sie bietet auch für Computerneulinge eine einfach benutzbare und vielseitig konfigurierbare Arbeitsoberfläche. Die WPS wurde zum ersten Mal mit OS/2 2.0, der ersten 32-Bit-Version des Betriebssystems, ausgeliefert und ist während der OS/2-Installation standardmäßig als Shell für das Betriebssystem vorgesehen.

WSoD: s. Bd. 2.

Die OS/2 Only! ist das anwenderorientierte OS/2-Magazin nur für OS/2 mit regelmäßigen Vorstellungen von OS/2-Software und Tests voll OS/2-tauglicher Hardware, ausführlichen Programmier- und Anwendungsworkshops, Treiber- und Fixpackunterstützung.

Das Magazin stellt Marktneuheiten vor, bietet Produktübersichten und gibt wichtige Tips & Tricks, die vor allem dem Endanwender nützen. Es bietet Hilfestellung bei Problemen, hilft beim Umsteigen auf das Betriebssystem OS/2 von anderen Plattformen und vermittelt in umfangreichen Artikeln und Workshops alles Nötige zum Programmieren in allen wichtigen Sprachen und zur Konfiguration von Netzwerken und Einzelplatzsystemen.

Außerdem in jeder Ausgabe: Ein umfangreiches Stichwortverzeichnis zum schnellen Auffinden von Informationen und ein Lexikon zur Erklärung aller wichtigen Begriffe.

CEF C.-E. Fischer Buchverlag. Stuttgart

ISSN 1436-8307